Route Optimization – Part 1

Cabrillo College

CIS 185 - CCNP 1

Rick Graziani and Homer Simpson

Cabrillo College

Spring 2006

Note to instructors

- If you have downloaded this presentation from the Cisco Networking Academy Community FTP Center, this may not be my latest version of this PowerPoint.
- For the latest PowerPoints for all my CCNA, CCNP, and Wireless classes, please go to my web site:

  http://www.cabrillo.edu/~rgraziani/

  - The username is *cisco* and the password is *perlman* for all of my materials.
- If you have any questions on any of my materials or the curriculum, please feel free to email me at graziani@cabrillo.edu (I really don't mind helping.) Also, if you run across any typos or errors in my presentations, please let me know.
- I will add "(Updated – *date*)" next to each presentation on my web site that has been updated since these have been uploaded to the FTP center.

Thanks! Rick

Route Optimization

- Passive Interfaces
- Route Filters
  - Distribute Lists
- Policy Routing
  - Route Maps
- Route Redistribution
  - Multiple Routing Protocols
  - Changing Administrative Distances
  - Default Metrics

Route Optimization

You can control when a router exchanges routing updates and what those updates.

You can also more tightly control the direction of network traffic

All by using:

- routing update controls
- policy-based routing
- route redistribution

Route Optimization

- Passive Interfaces
- Route Filters
  - Distribute Lists
- Policy Routing
  - Route Maps
- Route Redistribution
  - Multiple Routing Protocols
  - Changing Administrative Distances
  - Default Metrics

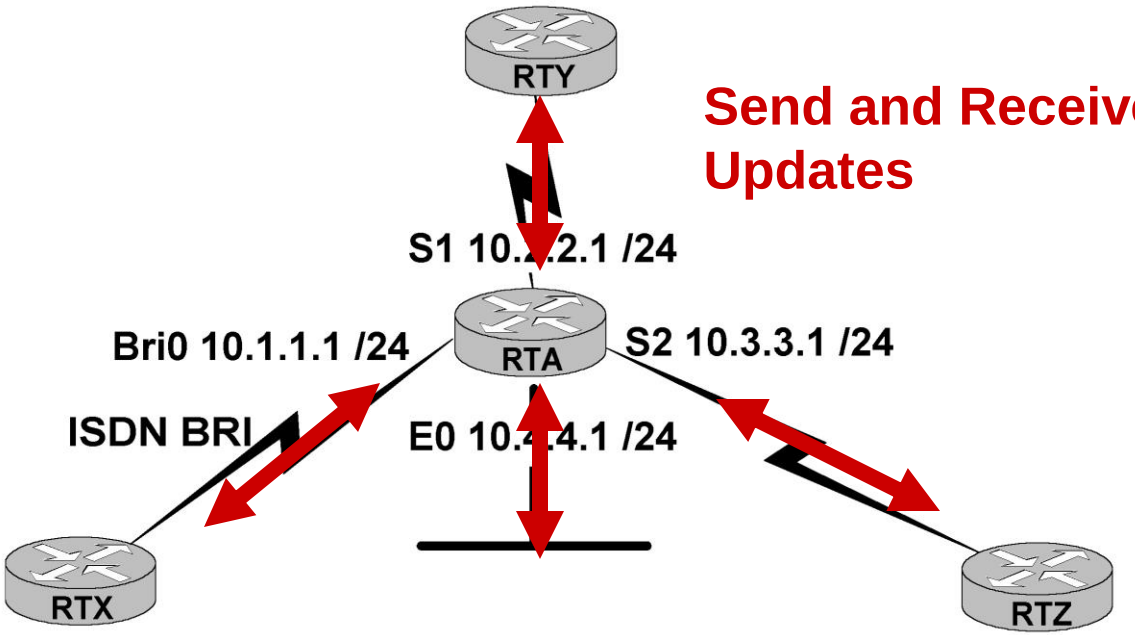
Passive Interfaces

The **`passive-interface`** command works differently with the different IP routing protocols that support it.

- RIP/IGRP:
  - Can receive updates but doesn't send.
- OSPF:
  - Routing information is neither sent nor received via a passive interface.
  - The network address of the passive interface appears as a stub network in the OSPF domain.
  - Better to use proper wildcard mask in network command.
- EIGRP:
  - The router stops sending hello packets on passive interfaces.
  - When this happens, the EIGRP router can't form neighbor adjacencies on the interface or send and receive routing updates.

Passive Interfaces


```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
```



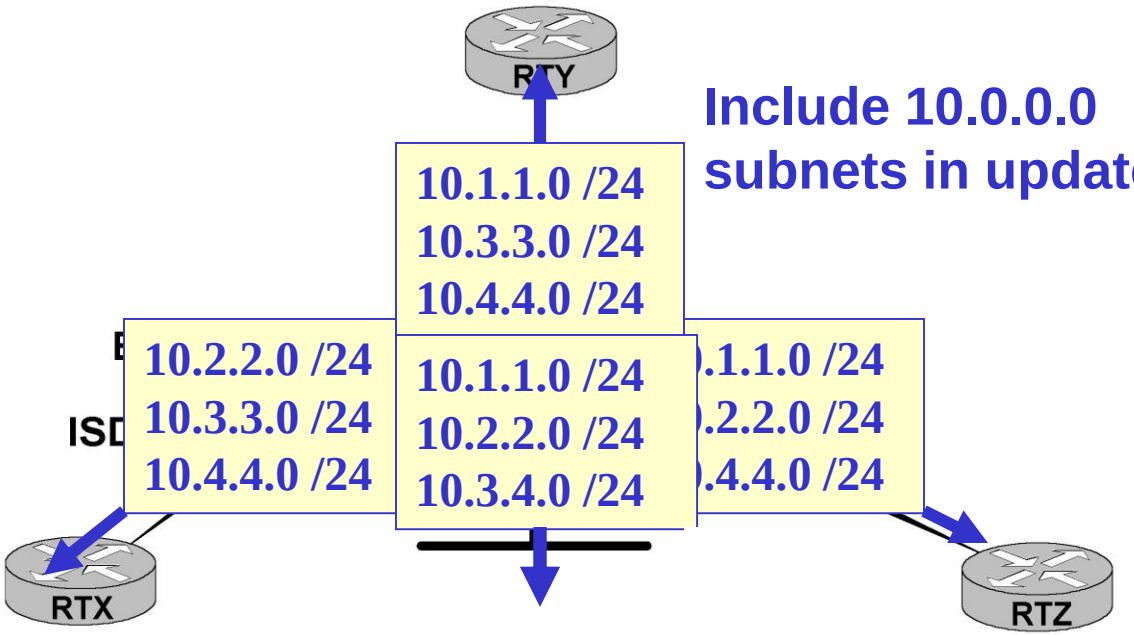
Send and Receive RIP Updates

By default:

- RIP updates are sent out all interfaces belonging to the 10.0.0.0 network.
- All directly connected subnets belonging to 10.0.0.0 network will be included in the RIP updates, plus any dynamically learned routes.

Passive Interfaces


```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
```



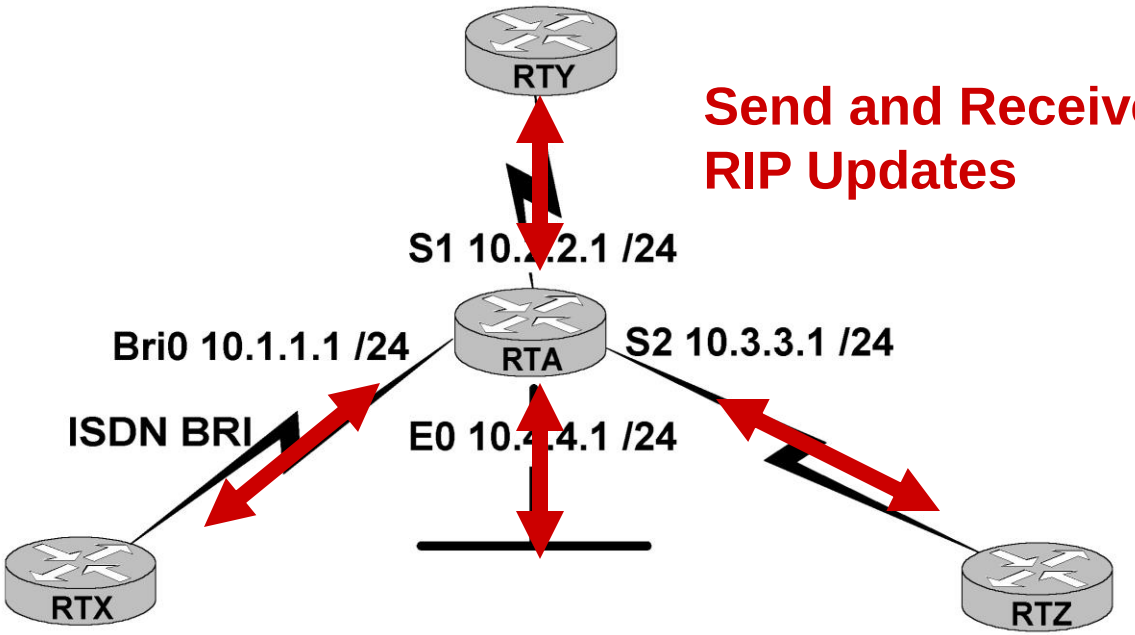
Include 10.0.0.0 subnets in updates

By default:

- RIP updates are sent out all interfaces belonging to the 10.0.0.0 network.
- All directly connected subnets belonging to 10.0.0.0 network will be included in the RIP updates, plus any dynamically learned routes.

Passive Interfaces


```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
```



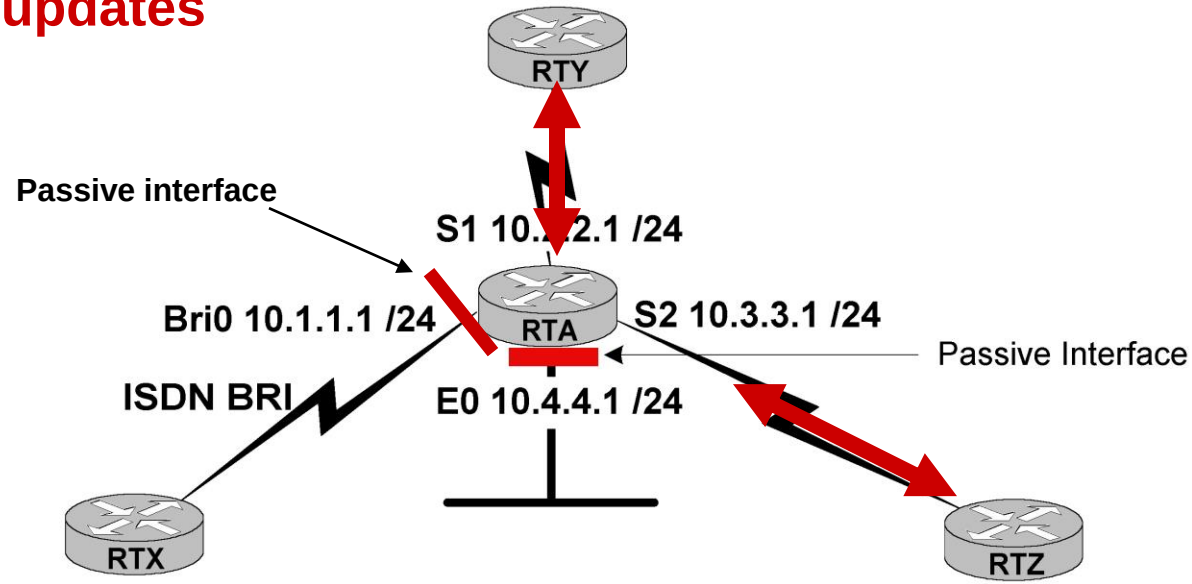
Default behavior maybe not the best:

- No need to send RIP updates out E0.
- RIP updates keeping the ISDN link up.

Passive Interfaces


```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#passive-interface e0
RTA(config-router)#passive-interface bri0
```

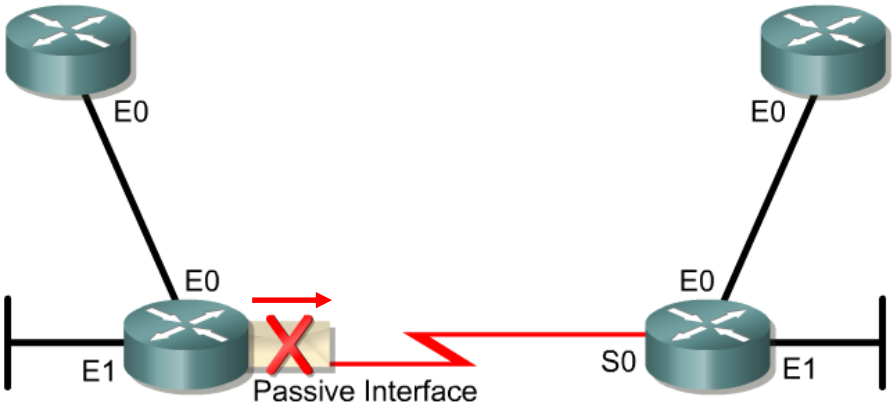
Passive Interfaces receive—but don't send--updates



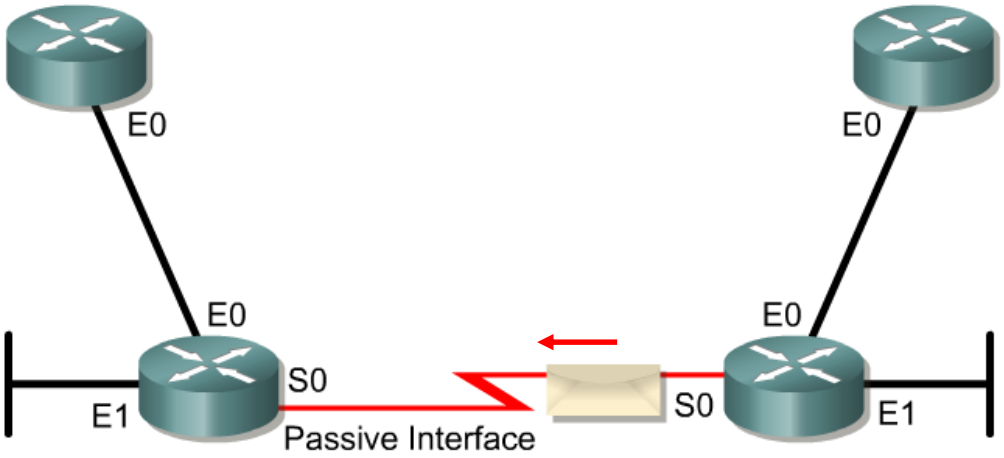
`passive-interface [default] {`*`interface-type number`*`}`

The default keyword sets all interfaces as passive by default.

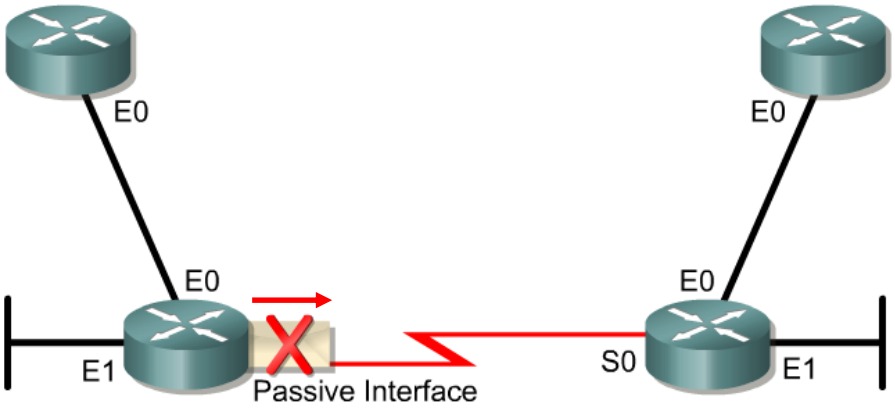
Passive Interfaces and RIP

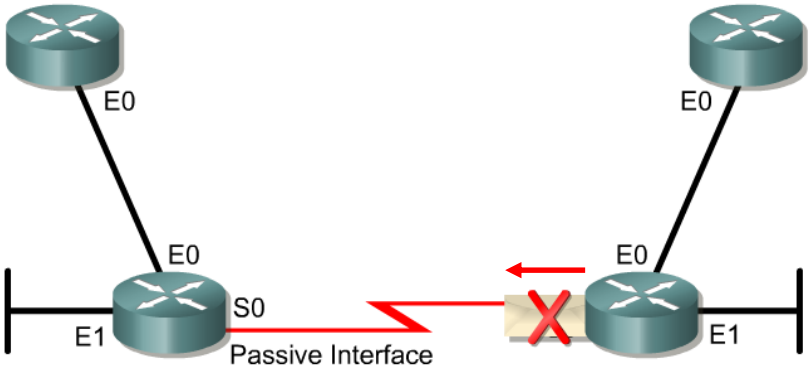


Passive Interfaces receive—but don't send--updates



Passive Interfaces and OSPF/EIGRP





The network address of the passive interface appears as a stub network in the OSPF domain.

Better to use proper wildcard mask in network command.

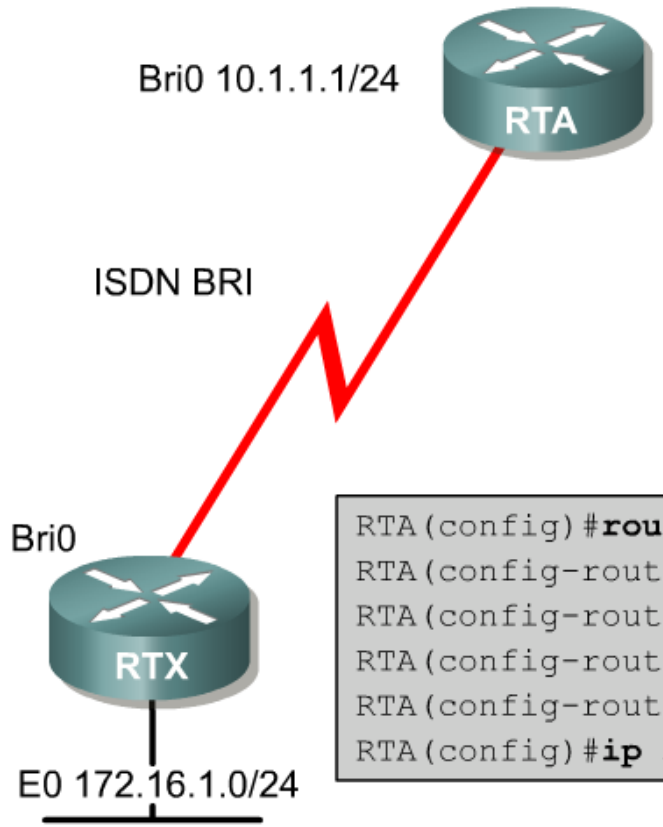
EIGRP router can't form neighbor adjacencies on the interface or send and receive routing updates. An EIGRP network can be configured as a stub network. (Doyle p.309)

Passive Interfaces and DDR

- You can use the **passive-interface** command on WAN interfaces to prevent routers from sending updates to link partners.
- There may be several reasons to squelch updates on the WAN.
  - If connected by a dial-on-demand ISDN link regular RIP updates will keep the link up constantly, and result in an eye-popping bill from the provider.

Passive Interfaces and DDR

- You can use the **passive-interface** command on WAN interfaces to prevent routers from sending updates to link partners.
- There may be several reasons to squelch updates on the WAN.
  - If connected by a dial-on-demand ISDN link regular RIP updates will keep the link up constantly, and result in an eye-popping bill from the provider.

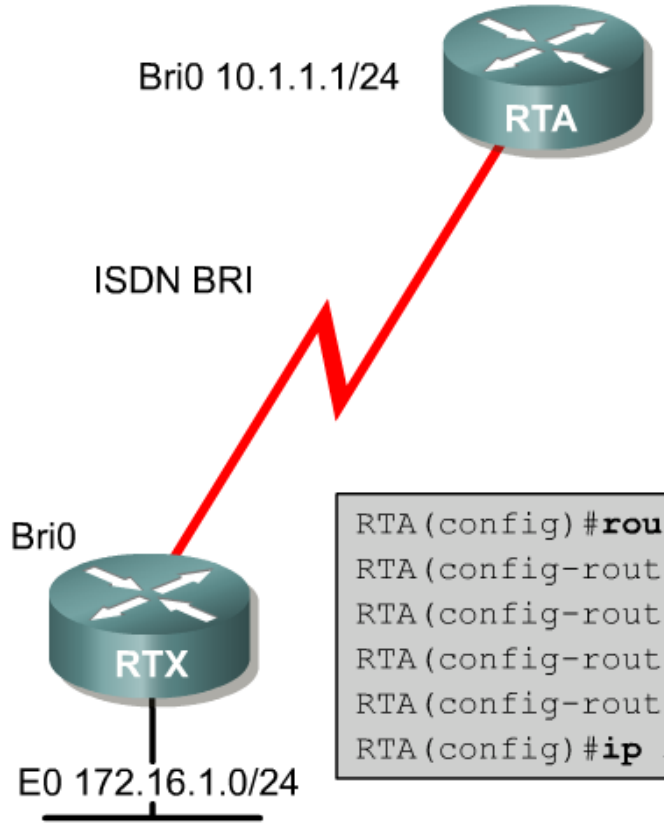



```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#passive-interface bri0
RTA(config-router)#redistributive static
RTA(config-router)#exit
RTA(config)#ip route 172.16.1.0 255.255.255.0 bri0
```

```
RTX(config)#ip route 0.0.0.0 0.0.0.0 bri0/0
```

Passive Interfaces

- The following example sets all interfaces as passive, then activates the Ethernet 0 interface:



RTA

```
router rip
    passive-interface default
    no passive-interface ethernet0
    network 10.0.0.0
```

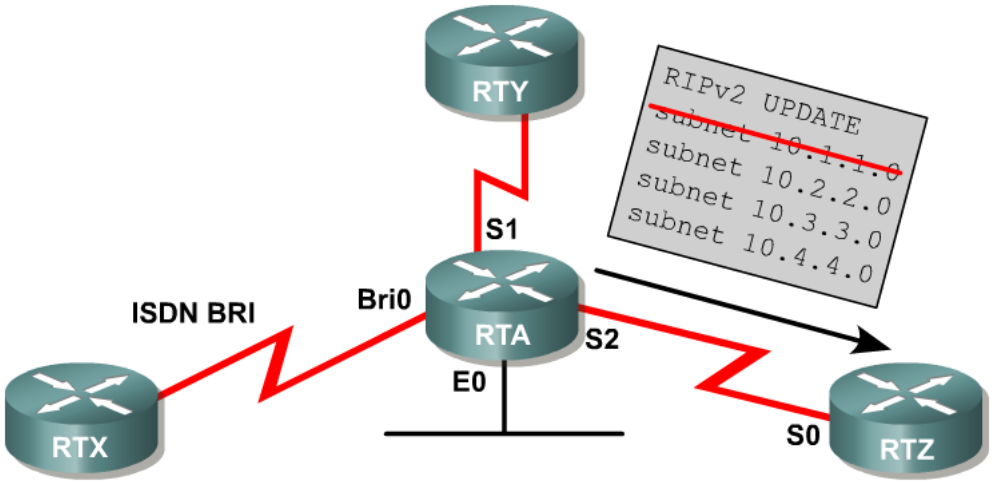
```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#passive-interface bri0
RTA(config-router)#redistributive static
RTA(config-router)#exit
RTA(config)#ip route 172.16.1.0 255.255.255.0 bri0
```

```
RTX(config)#ip route 0.0.0.0 0.0.0.0 bri0/0
```

Route Optimization

- Passive Interfaces
- Route Filters
  - Distribute Lists
- Policy Routing
  - Route Maps
- Route Redistribution
  - Multiple Routing Protocols
  - Changing Administrative Distances
  - Default Metrics

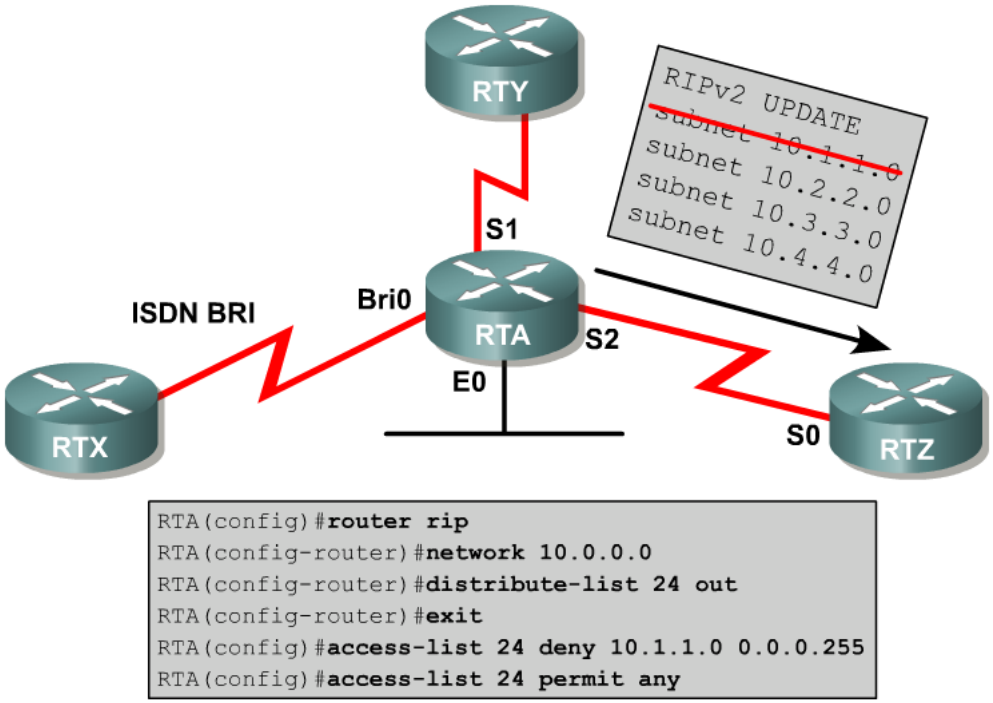
Route Filters




```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out
RTA(config-router)#exit
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

- Configuring an **passive interface** prevents it from sending updates entirely, but there are times when you need to suppress only certain routes in the update from being sent or received.
- We can use a **distribute-list** command to pick and choose what routes a router will send or receive updates about.
- The **distribute-list** references an **access-list**, which creates a **route filter**
  - **Route filter**– a set of rules that precisely **controls what routes a router sends or receives in a routing update**.

Route Filters



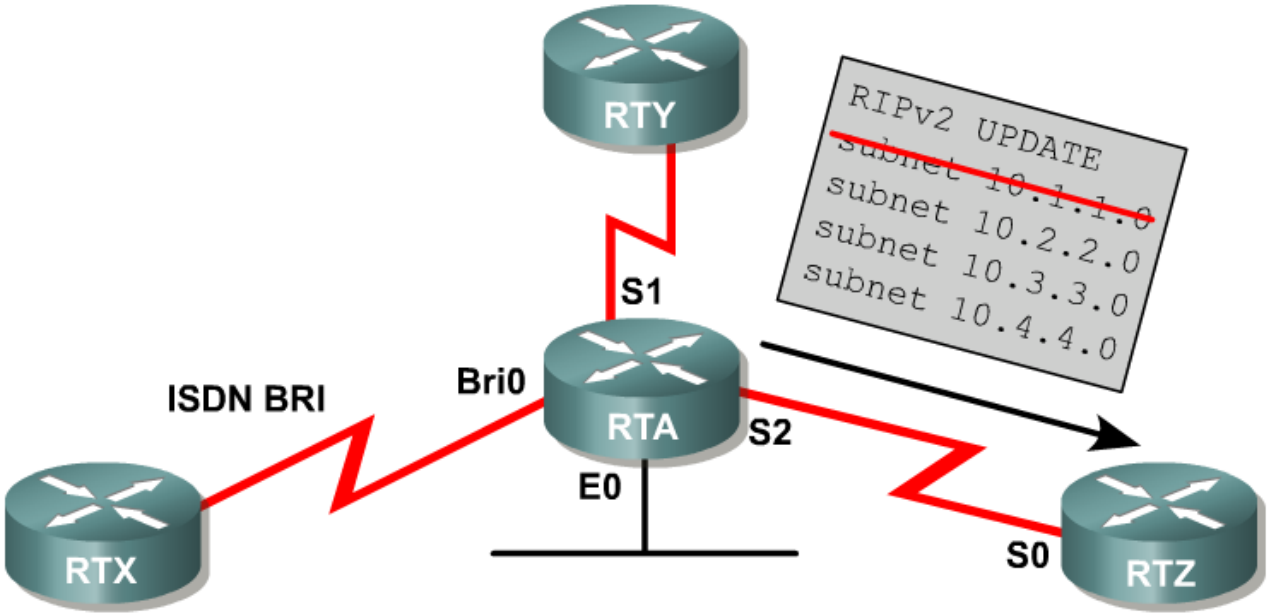

```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out
RTA(config-router)#exit
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

Route filters may be needed to enforce a routing policy that's based on some external factor such as

- link expense
- administrative jurisdiction
- security concerns
- overhead reduction—prevents access routers from receiving the complete (and possibly immense) core routing table

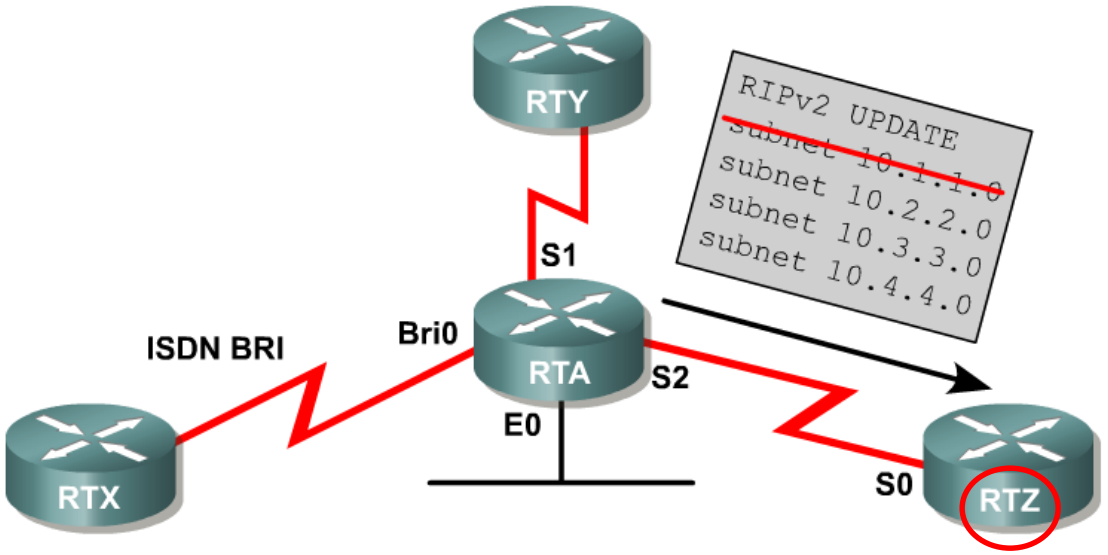
Route Filters – Inbound and Outbound

Let's take a look on how keep subnet 10.1.1.0 from entering RTZ!




```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out
RTA(config-router)#exit
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

Route Filters - Inbound



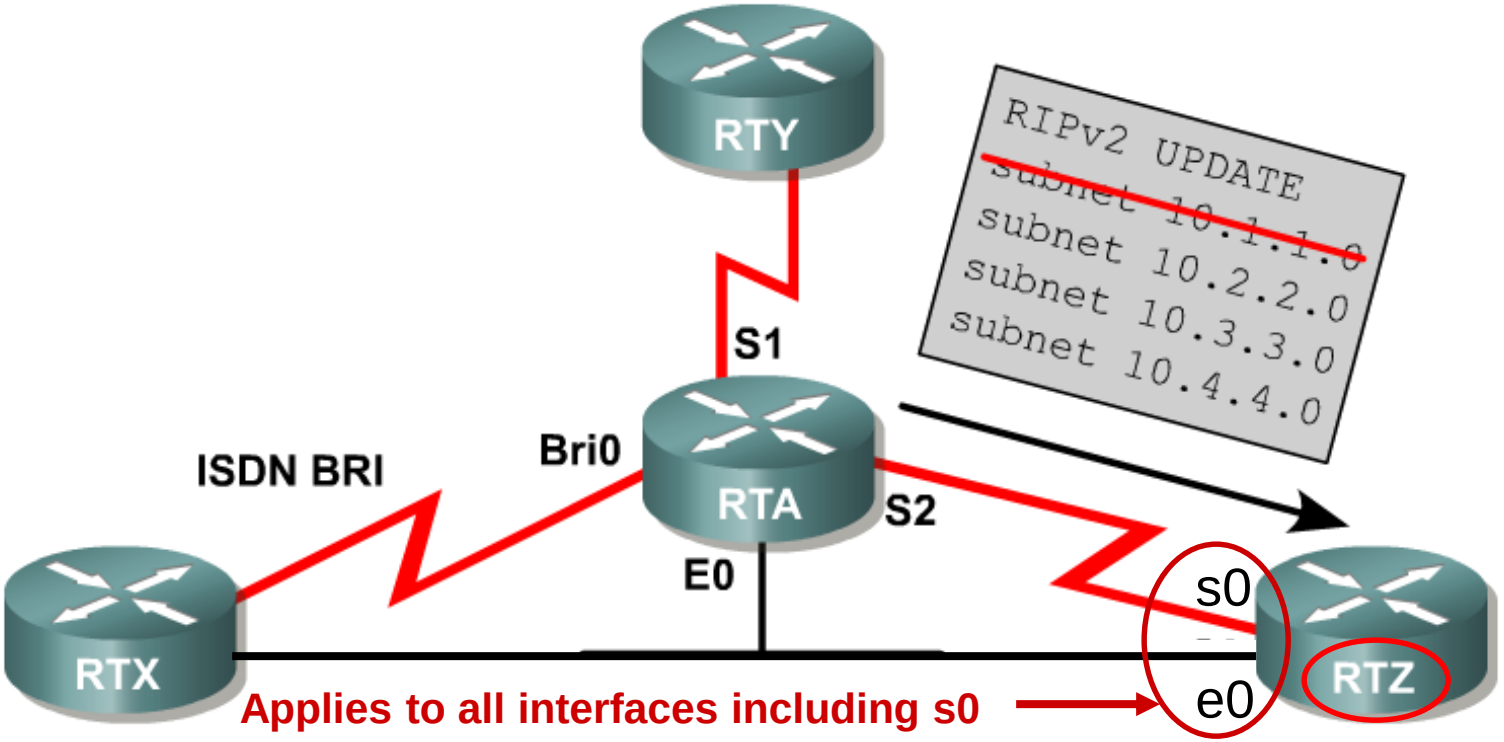
Inbound interfaces:

- When applied to inbound updates, the syntax for configuring a route filter is as follows:

```
Router(config-router)#distribute-list access-list-number in
   [interface-name]
```

Note: This does not permit/deny packets from entering the routers, only what routes a router will send or receive updates about.

Route Filters Inbound - Global

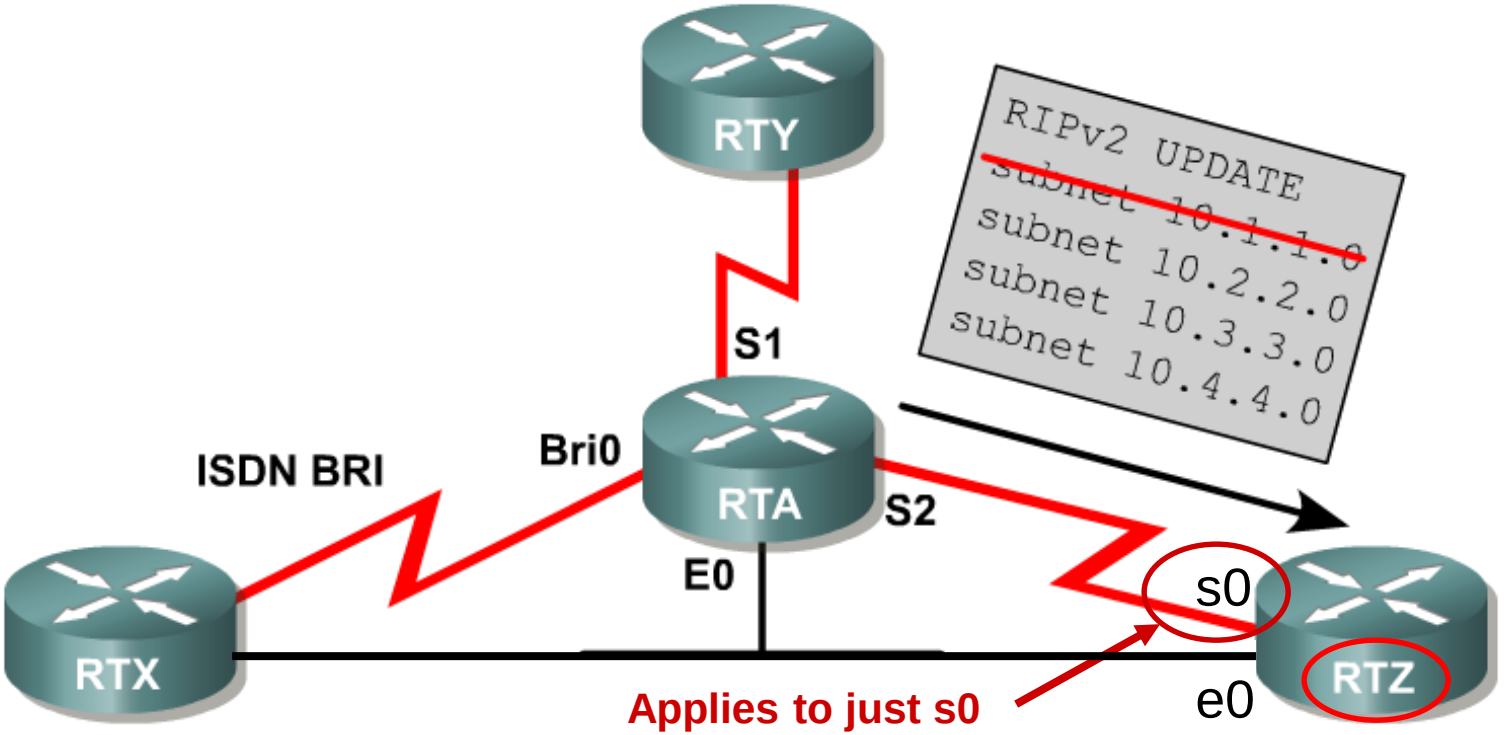


RTZ

```
(config)#router rip
(config-router)#network 10.0.0.0
(config-router)#distribute-list 16 in
(config-router)#exit
(config)#access-list 16 deny 10.1.1.0 0.0.0.255
(config)#access-list 16 permit any
```

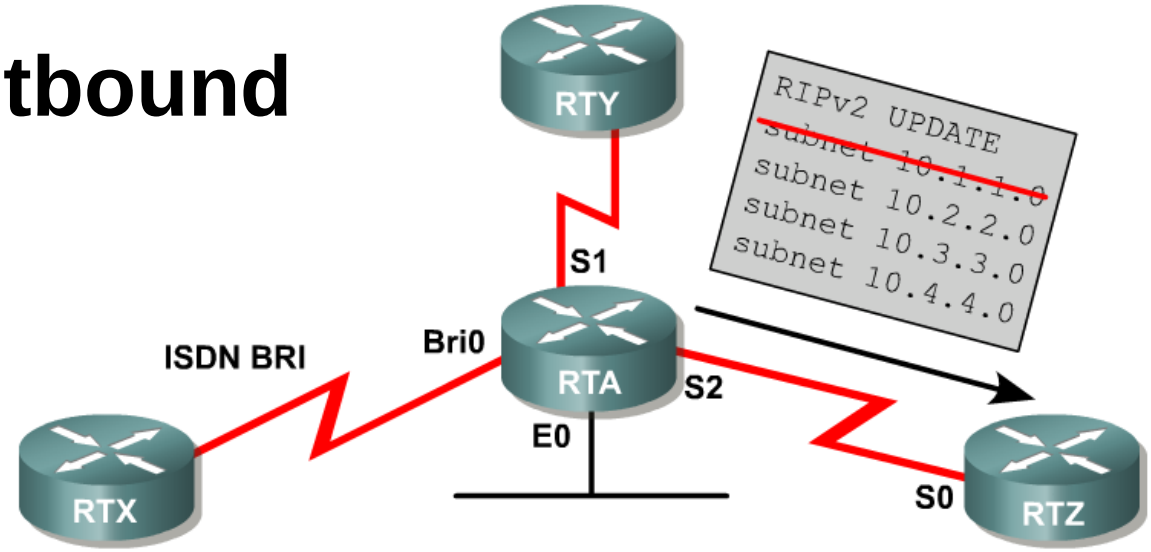
Graphic incorrect in curriculum

Route Filters Inbound - Interface




```
RTZ(config)#router rip
RTZ(config-router)#network 10.0.0.0
RTZ(config-router)#distribute-list 16 in s0
RTZ(config)#access-list 16 deny 10.1.1.0 0.0.0.255
RTZ(config)#access-list 16 permit any
```

Route Filters Outbound



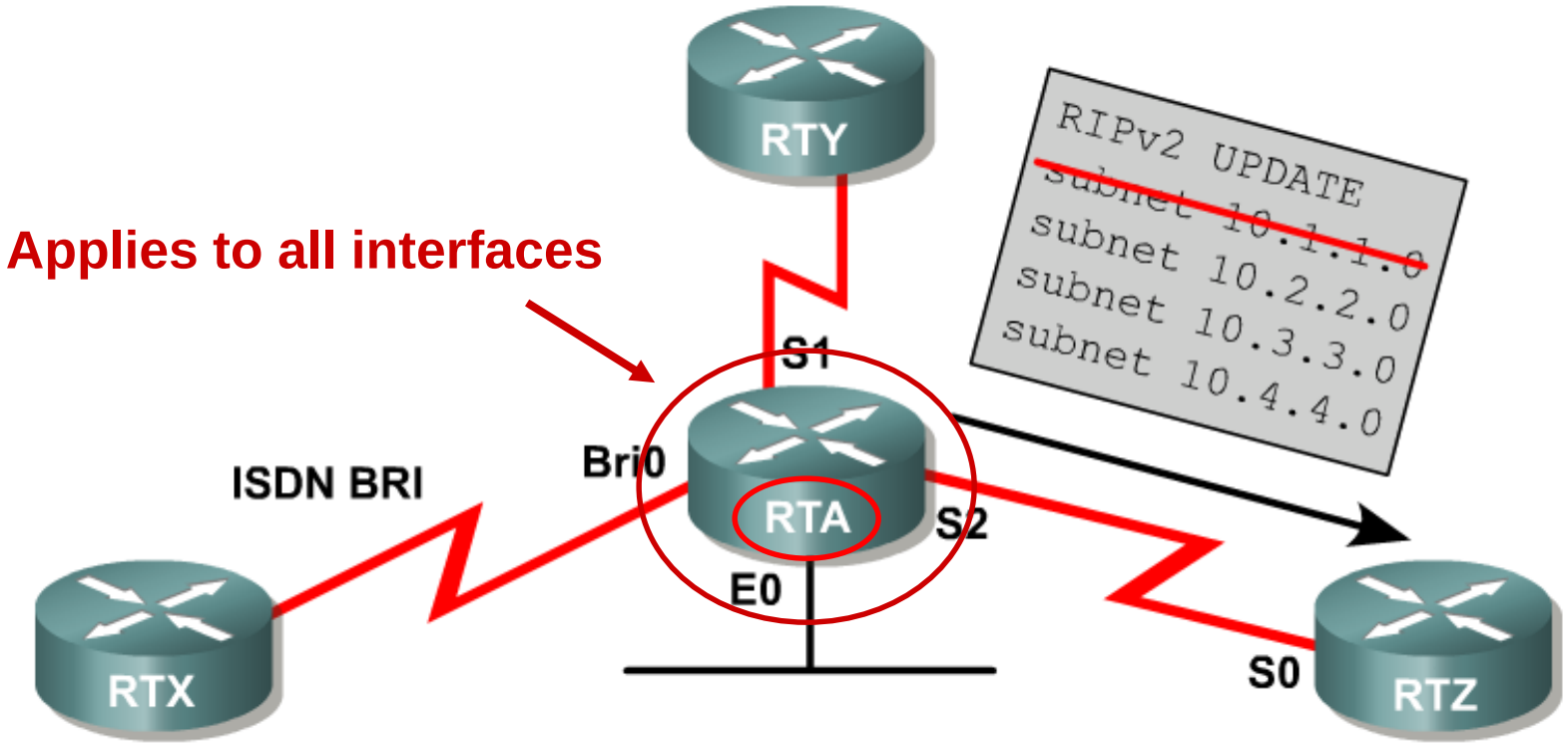

```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out
RTA(config-router)#exit
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

Outbound interfaces:

- When applied to outbound updates, the syntax can be more complicated:

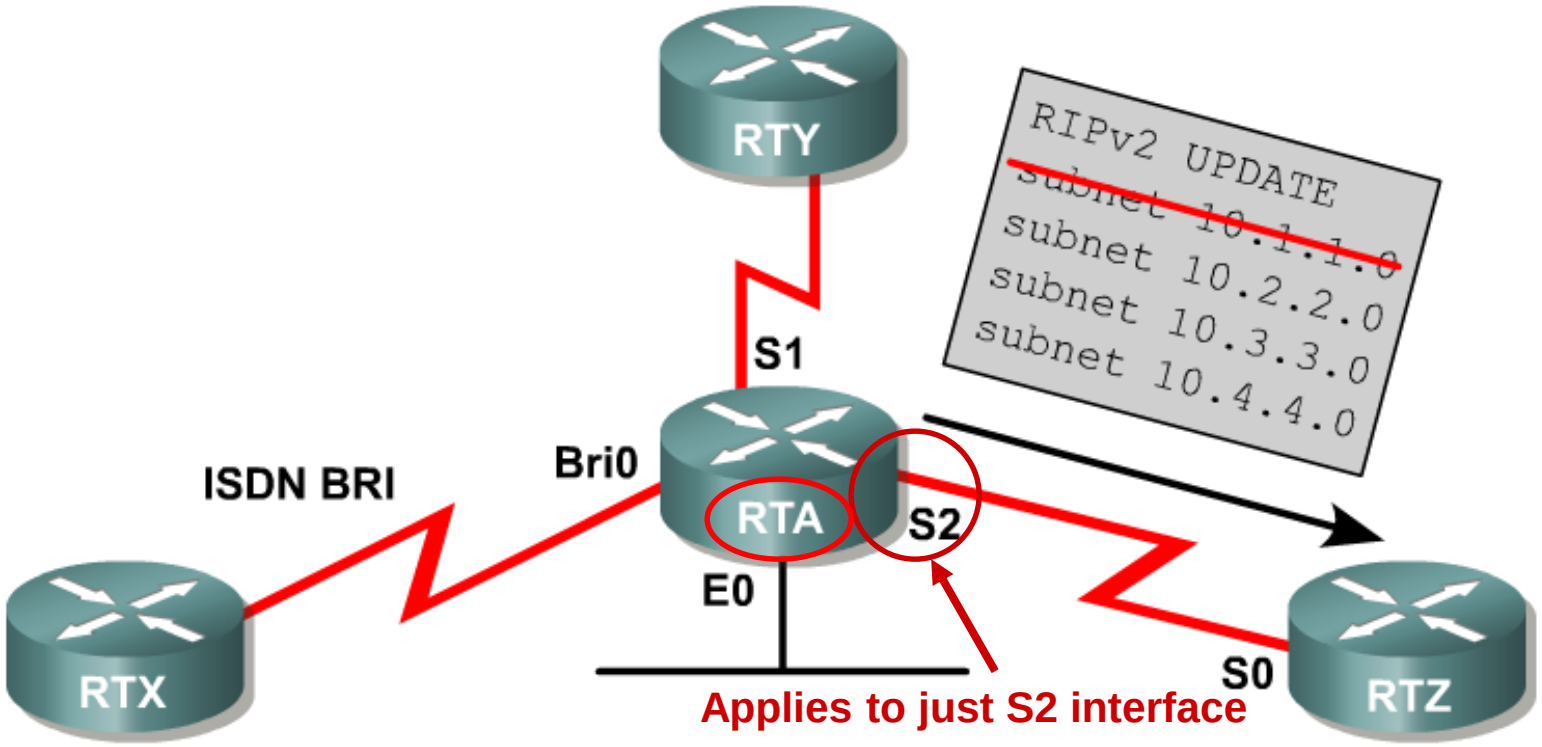
```
Router(config-router)#distribute-list access-list-number out
   [interface-name | routing-process | as-number]
```

Route Filters Outbound - Global




```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out
RTA(config-router)#exit
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

Route Filters Outbound - Interface




```
RTA(config)#router rip
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 24 out s2
RTA(config)#access-list 24 deny 10.1.1.0 0.0.0.255
RTA(config)#access-list 24 permit any
```

Route Filters

For each interface and routing process, Cisco IOS permits:

- one incoming global distribute-list
- one outgoing global distribute-list
- one incoming interface distribute-list
- one outgoing interface distribute-list

```
RTZ(config)#router rip
RTZ(config-router)#distribute-list 1 in
RTZ(config-router)#distribute-list 2 out
RTZ(config-router)#distribute-list 3 in e0
RTZ(config-router)#distribute-list 4 out e0
```

Route Filters - Verification


```
RTZ(config)#router rip
RTZ(config-router)#distribute-list 1 in
RTZ(config-router)#distribute-list 2 out
RTZ(config-router)#distribute-list 3 in e0
RTZ(config-router)#distribute-list 4 out e0

RTZ#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 25 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is 2
    Ethernet0 filtered by 4
  Incoming update filter list for all interfaces is 1
    Ethernet0 filtered by 3
```

Route Filters and Link State Routing Protocols

- Routers running link state protocols determine their routes based on information in their link state database, rather than the advertised route entries of its neighbors.
- **Route filters have no effect** on link state advertisements or the link state database.
  - Remember, a basic requirement of link state routing protocols is that routers in an area must have identical link state databases.
- A route filter can influence the route table of the router on which the filter is configured, but has no effect on the route entries of neighboring routers.
- Route filters are mainly used at redistribution points, such as on an ASBR. (Part 2).

Route Filters and Link State Routing Protocols (FYI)

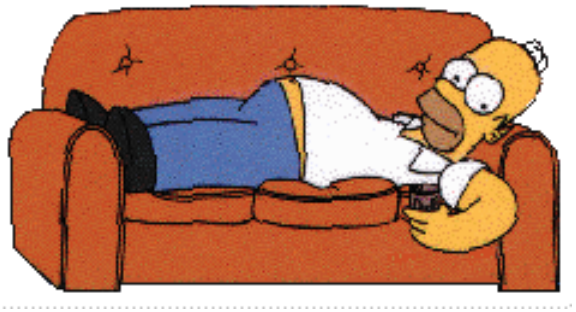
Q: Can I use the distribute-list in/out command with OSPF to filter routes?

A: OSPF routes **cannot** be filtered from entering the OSPF database. The **distribute-list in** command only filters routes from entering the routing table, but it doesn't prevent link-state packets from being propagated.

- The command **distribute-list out** works only on the routes being redistributed by the autonomous system boundary routers (ASBRs) into OSPF.
- It can be applied to external type 2 and external type 1 routes, but not to intra-area and inter-area routes.

“Passive” EIGRP interfaces

- A **passive interface <u>cannot</u>** send EIGRP hellos, which thus prevents adjacency relationships with link partners.
- An administrator can create a “psuedo” passive EIGRP interface by using a **route filter** that suppresses *all* routes from the EIGRP routing update.



```
RTA(config)#router eigrp 364
RTA(config-router)#network 10.0.0.0
RTA(config-router)#distribute-list 5 out s0
RTA(config-router)#exit
RTA(config)#access-list 5 deny any
```

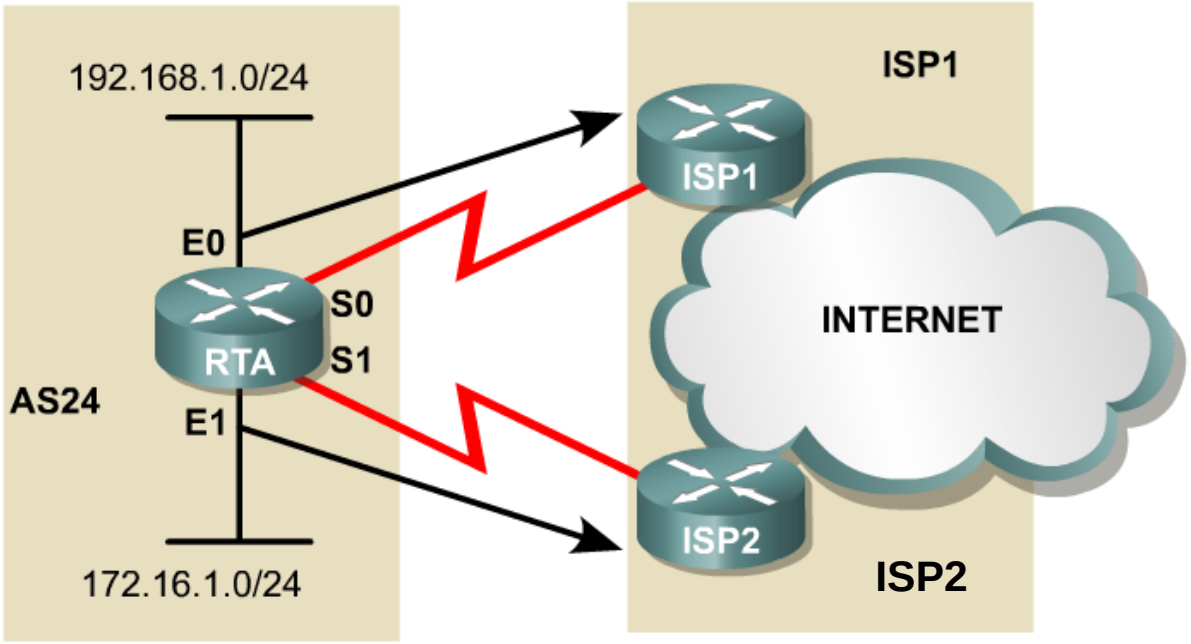
Route Optimization

- Passive Interfaces
- Route Filters
  - Distribute Lists
- Policy Routing
  - Route Maps
- Route Redistribution
  - Multiple Routing Protocols
  - Changing Administrative Distances
  - Default Metrics

Policy Routing

- **Static routes**: You can use the **ip route** command to dictate which path a router will select to a given destination, based on the destination address..
- However, through **policy routing**, you can manually program a router to *choose a route* based not only on destination, but on source as well.
- Human factors such as monetary expense, organizational jurisdiction, or security issues can lead administrators to establish ***policies***, or rules that routed traffic should follow.
- Left to their default behavior, routing protocols may arrive at path decisions that conflict with these policies.
- **Policy routes** are nothing more than ***sophisticated static routes***.

Policy Routing

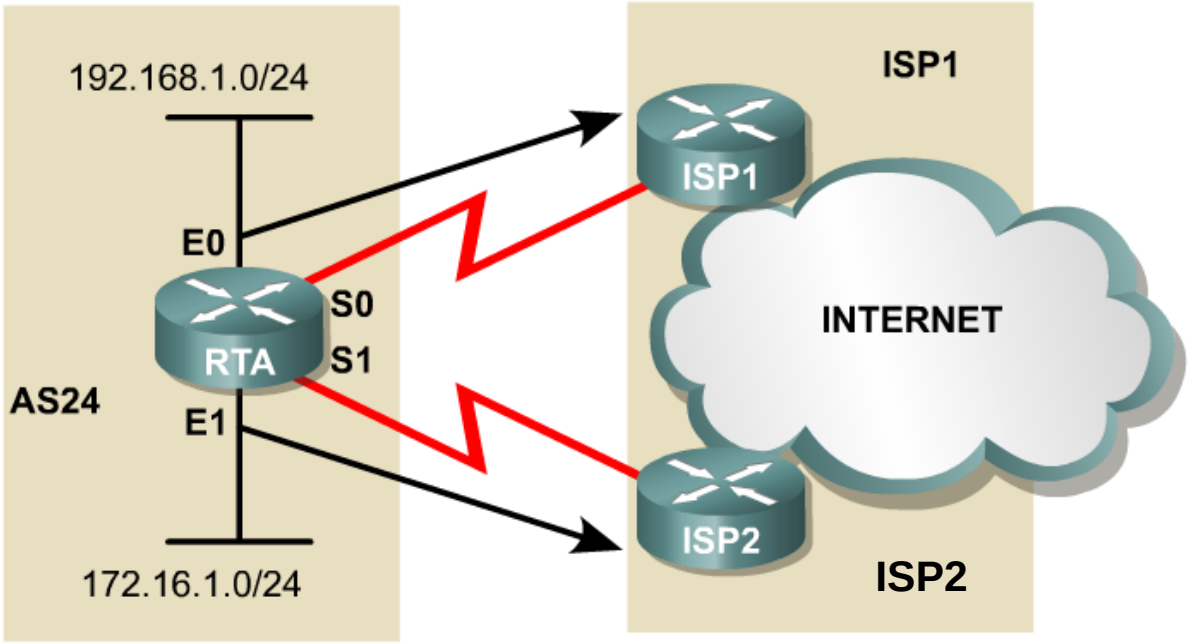


- Policy routing is used to:
  - Override dynamic routing
  - Take precise control of how their routers handle certain traffic.
- Although policy routing can be used to control traffic within an AS, it is typically used to control routing between autonomous systems (ASs).
  - Policy routing is used extensively with exterior gateway protocols (EGPs), such as BGP.
  - Later

Policy Routing

~~Routing Table~~

Routing Policy



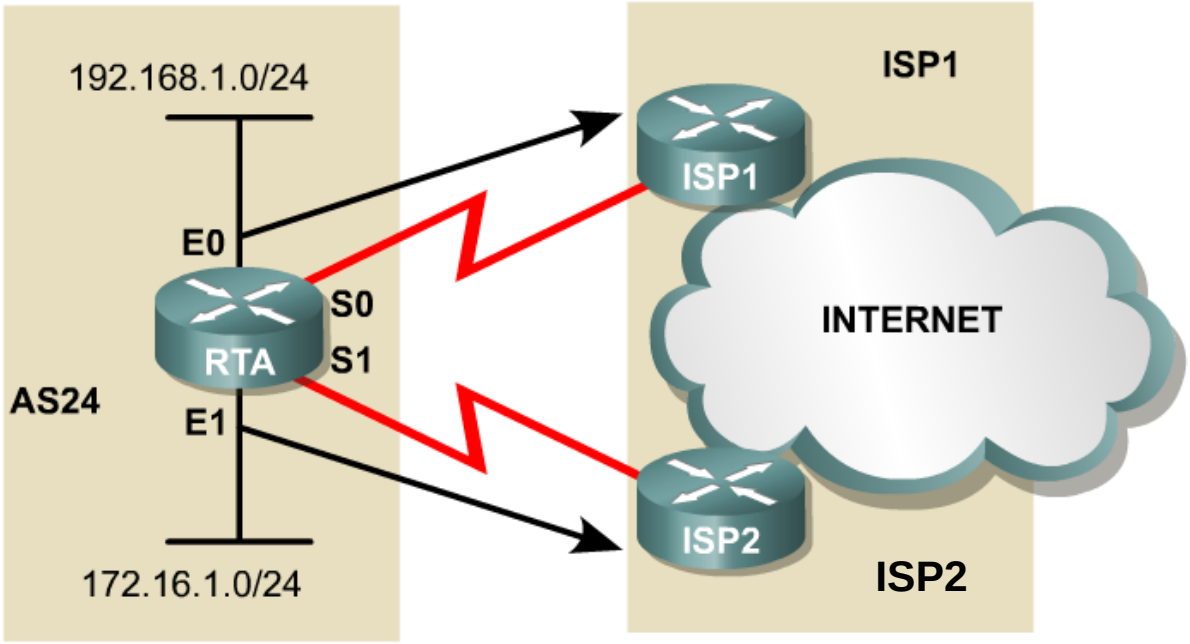
- The **route-map** command is used to configure policy routing, which is often a complicated task.

- **"Policy routing is a technique that allows the administrator to instruct the router to forward packets according to a configured policy rather than according to the contents of the routing table and the packet's destination address." Alex Zinin**

Policy Routing

~~Routing Table~~

Routing Policy



- A route map is defined using the following syntax:

```
Router(config)# route-map map-tag [permit | deny] [sequence-number]
Router(config-map-route)#
```

- Default is permit.
- Deny is more often used with route maps and redistribution. (later)
- You can use the optional ***sequence-number*** to indicate the position a new route map is to have in the list of route maps already configured with the same name.
- If you don't specify a sequence number, the first route map condition will be automatically numbered as ***10***.

Policy Routing

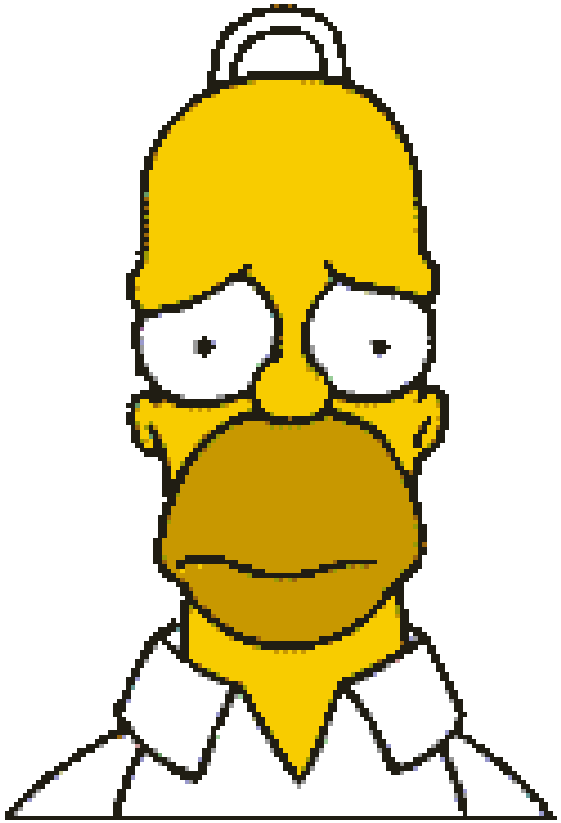
```
RTA(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RTA(config)#access-list 2 permit 172.16.1.0 0.0.0.255
RTA(config)#route-map ISP1 permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set interface serial0/0
RTA(config-route-map)exit
RTA(config)#route-map ISP2 permit 20
RTA(config-route-map)#match ip address 2
RTA(config-route-map)#set interface serial0/1
```

Once you have entered the **route-map** command, you can enter **set** and **match** commands in the route-map configuration mode.

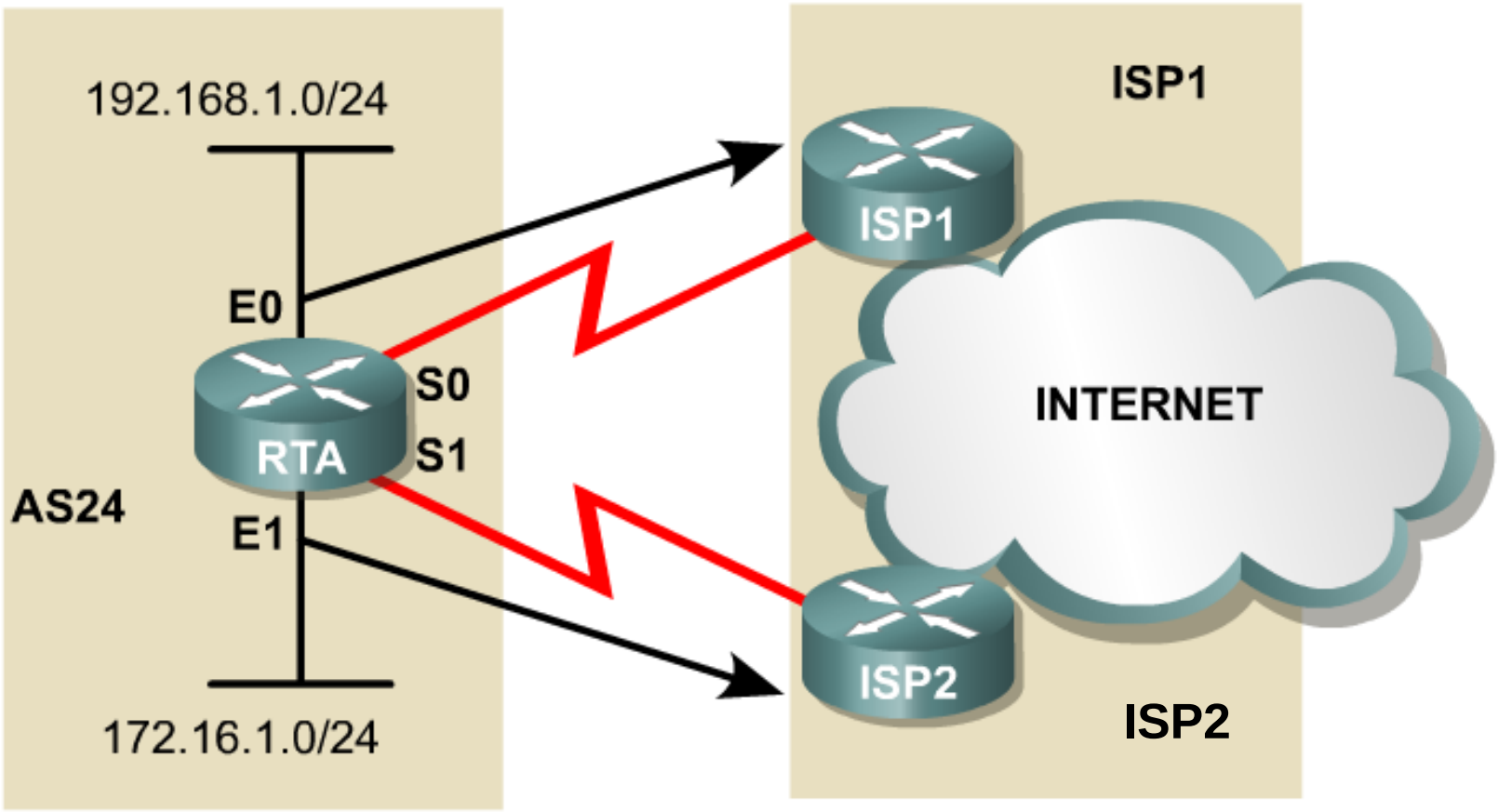
- Each **route-map** command has a list of **match** and **set** commands associated with it.
- The **match** commands specify the *match criteria*—the conditions that should be tested to determine whether or not to take action.
- The **set** commands specify the *set actions*—the actions to perform if the match criteria are met.

Policy Routing

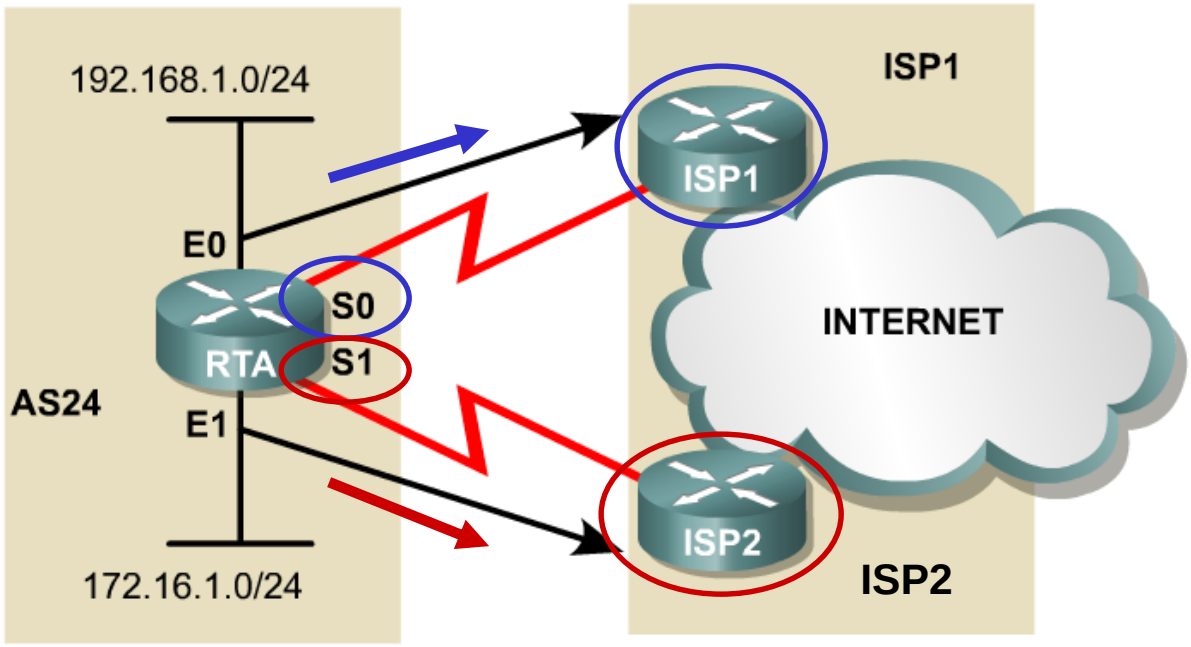
Don’t worry, several examples will help show how this works…



Policy Routing Example



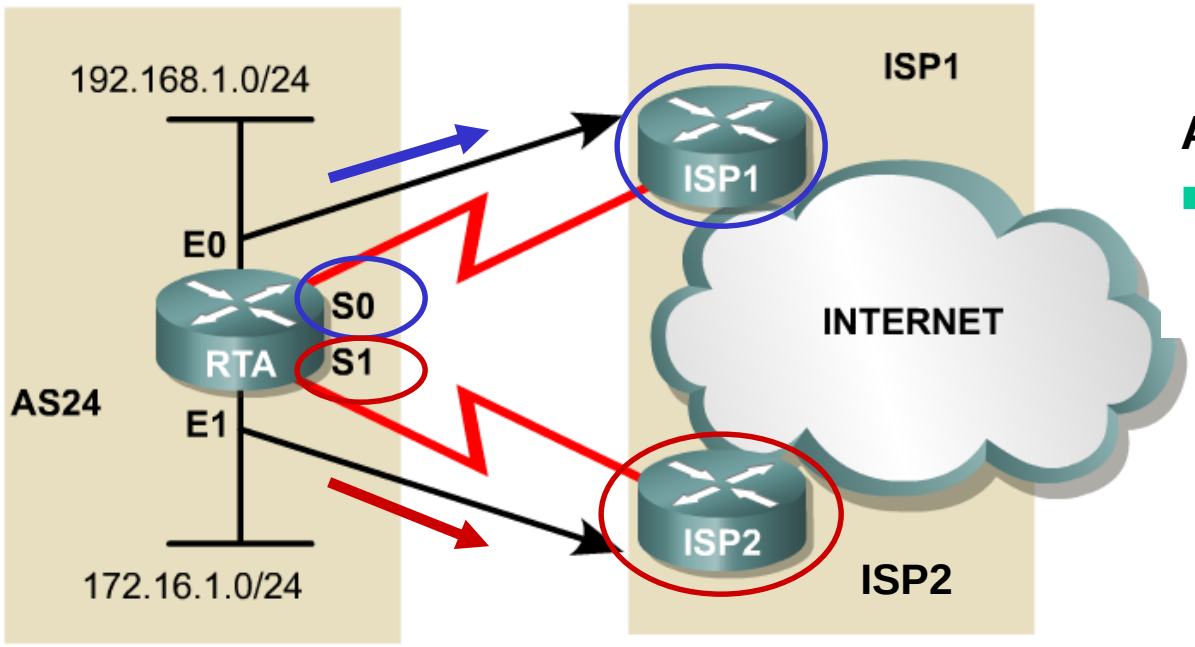
Policy Routing Example



Assume for this example that the policy we want to enforce is this:

- Internet-bound traffic from 192.168.1.0 /24 is to be routed to ISP1
- Internet-bound traffic from 172.16.1.0 /24 is to be routed to ISP2.

Policy Routing Example



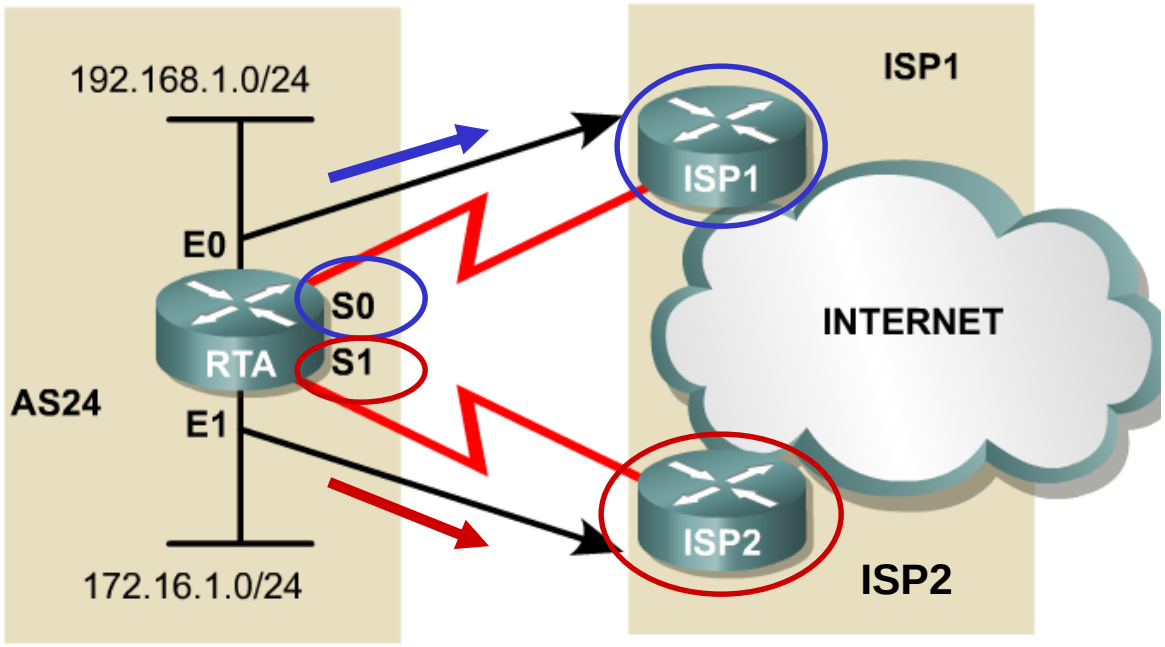
Cabrillo College

Access Lists

- First we configure two **access lists** with these commands:

```
RTA(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RTA(config)#access-list 2 permit 172.16.1.0 0.0.0.255
```

Policy Routing Example

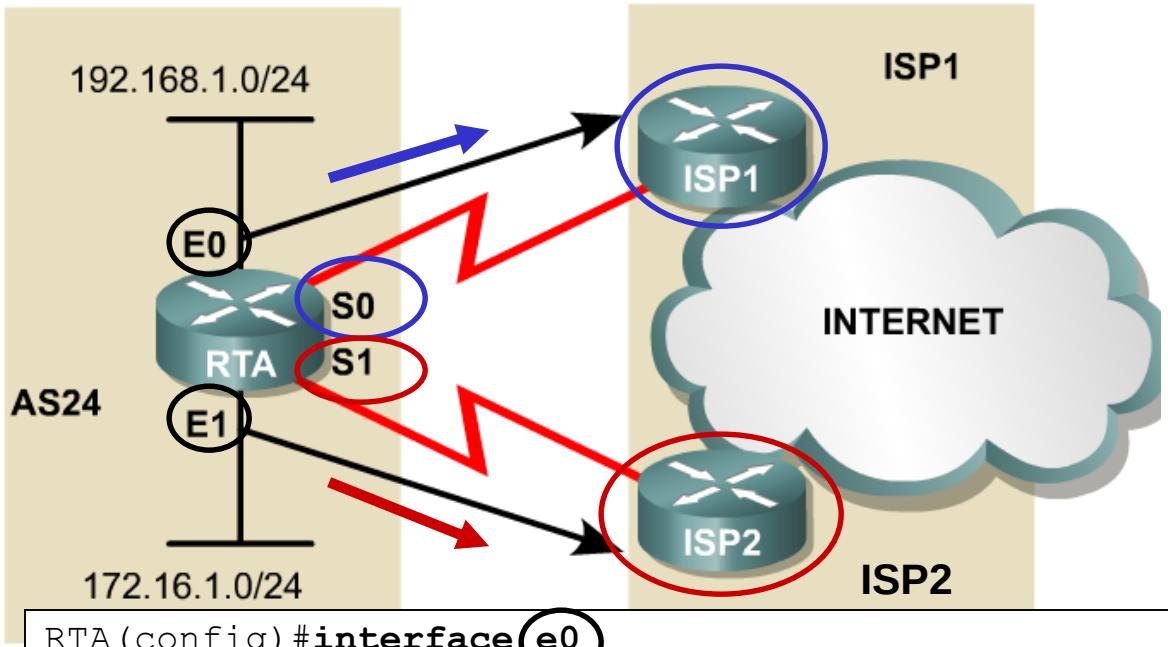


Global: route-maps

- Next we configure two **policies** with these commands:
  - The ISP1 route map will match access-list 1, and route traffic out S0 toward ISP1.
  - The ISP2 route map will match access-list 2, and route that traffic out S1 toward ISP2.
- More later on match and set

```
RTA(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RTA(config)#access-list 2 permit 172.16.1.0 0.0.0.255
RTA(config)#route-map ISP1 permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set interface s0
RTA(config)#route-map ISP2 permit 10
RTA(config-route-map)#match ip address 2
RTA(config-route-map)#set interface s1
```

Policy Routing Example

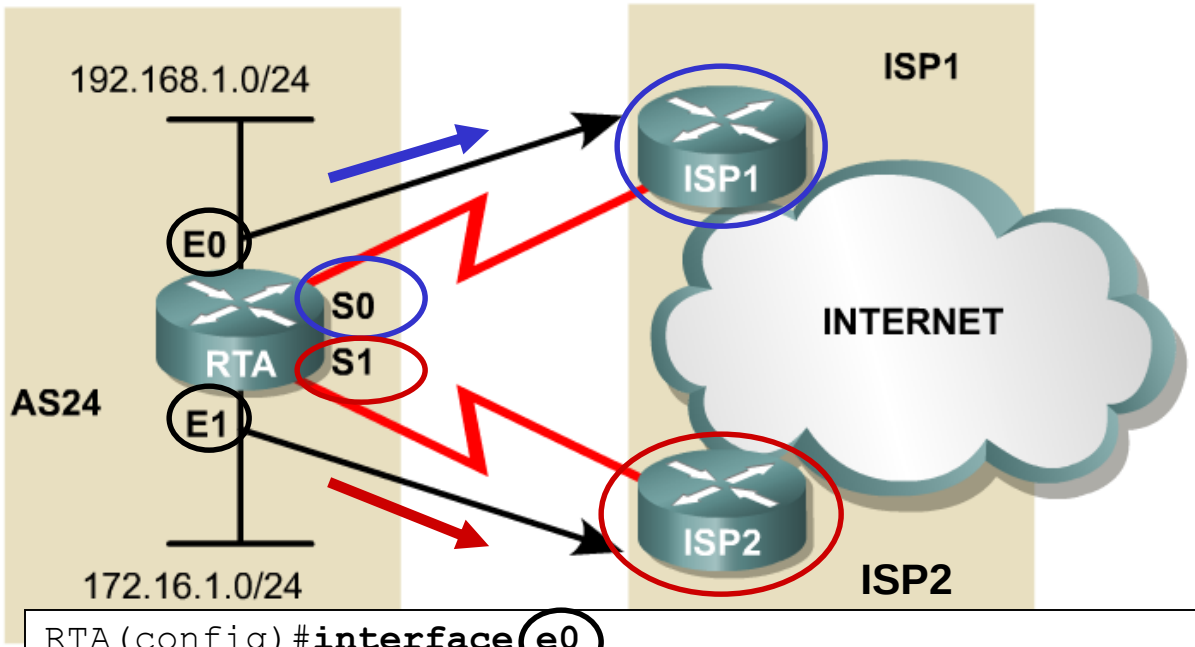


Interface: policy route-maps

- The final step is to apply each route map to the appropriate interface on RTA using the **ip policy route-map** command.
- **ip policy route-map *map-tag***
- With the route maps applied to the appropriate LAN interfaces, we have successfully implemented policy routing.

```
RTA(config)#interface e0
RTA(config-if)#ip policy route-map ISP1
RTA(config)#interface e1
RTA(config-if)#ip policy route-map ISP2
RTA(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RTA(config)#access-list 2 permit 172.16.1.0 0.0.0.255
RTA(config)#route-map ISP1 permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set interface s0
RTA(config)#route-map ISP2 permit 10
RTA(config-route-map)#match ip address 2
RTA(config-route-map)#set interface s1
```

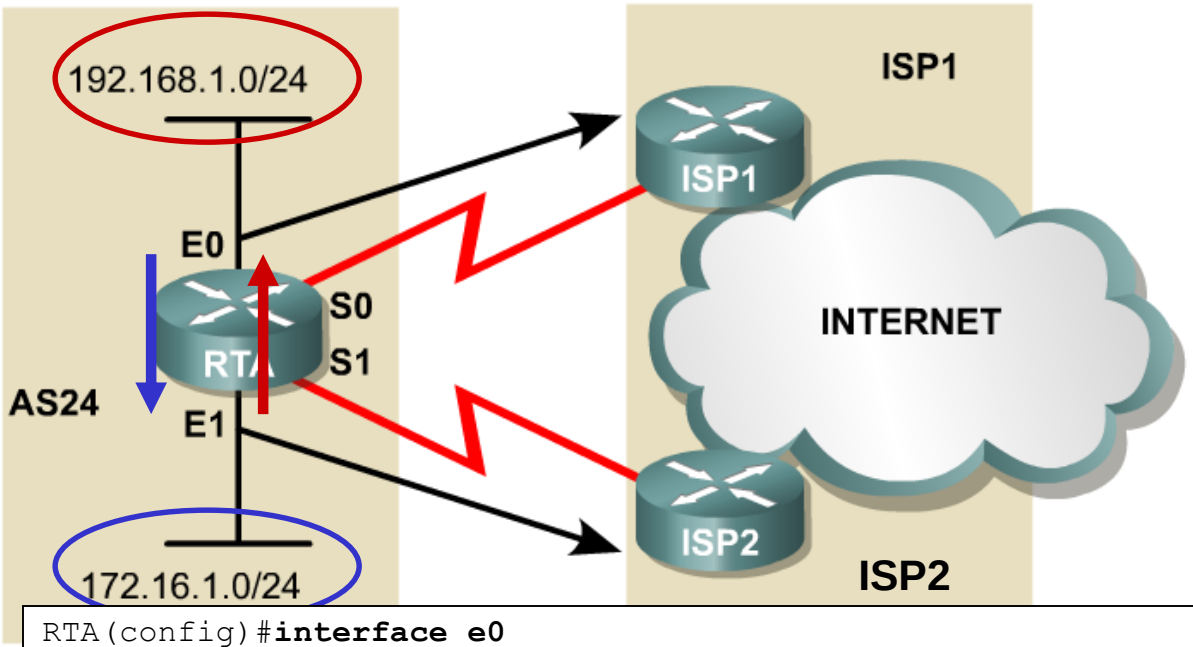
Policy Routing Example



- With the route maps applied to the appropriate **incoming** LAN interfaces, we have successfully implemented policy routing.
- **Note 1**: All other traffic will be routed normally according to their destination address.

```
RTA(config)#interface e0
RTA(config-if)#ip policy route-map ISP1
RTA(config)#interface e1
RTA(config-if)#ip policy route-map ISP2
RTA(config)#access-list 1 permit 192.168.1.0 0.0.0.255
RTA(config)#access-list 2 permit 172.16.1.0 0.0.0.255
RTA(config)#route-map ISP1 permit 10
RTA(config-route-map)#match ip address 1
RTA(config-route-map)#set interface s0
RTA(config)#route-map ISP2 permit 10
RTA(config-route-map)#match ip address 2
RTA(config-route-map)#set interface s1
```

Policy Routing Example



Note 2: What about traffic between 172.16.1.0 and 192.168.1.0?

- In this case, they will not be able to communicate.
- If there was a route for those networks on ISP1 and ISP2, then traffic would be routed from RTA to ISP1/ISP2 and back to RTA for the other LAN network.
- Fix? Use extended access lists and add a previous route-map statement that sends traffic to the other LAN out the other Ethernet interface.

```
RTA(config)#interface e0
RTA(config-if)#ip policy route-map ToNet172
RTA(config-if)#ip policy route-map ISP1
RTA(config)#interface e1
RTA(config-if)#ip policy route-map ToNet192
RTA(config-if)#ip policy route-map ISP2
RTA(config)#access-list 101 permit 192.168.1.0 0.0.0.255 172.16.1.0 0.0.0.255
RTA(config)#access-list 102 permit 172.16.1.0 0.0.0.255 192.168.1.0 0.0.0.255
RTA(config)#route-map ToNet172 permit 10
RTA(config-route-map)#match ip address 101
RTA(config-route-map)#set interface e1
RTA(config)#route-map ToNet192 permit 10
RTA(config-route-map)#match ip address 102
RTA(config-route-map)#set interface e0
```

Another Policy Routing Example

Jeff Doyle, Routing TCP/IP Vol. I

- **Policy routes** are nothing more than *sophisticated static routes*.
- **Static routes** forward a packet to a specified next hop based on ***destination address*** of the packet.
- **Policy routes** forward a packet to a specified next hop based on the ***source of the packet***.
  - Policy routes can also be linked to extended IP access lists so that routing may be based on ***protocol types and port numbers***.
  - Like a static route, *policy route influences the routing only of the router on which it is configured*.

Match Options (a sample)

- `Router(config-route-map)#` **`match length`** ***`min max`***
  - Matches the Layer 3 length of the packet.

- `Router(config-route-map)#` **`match ip address {`*`access-list-number`*** **`|`** ***`name`***`}` **`[`***`...access-list-number`*** **`|`** ***`name`***`]`
  - Matches the source and destination IP address that is permitted by one or more standard or extended access lists.

- **If you do not specify a match command, the route map applies to all packets.**

Set Options (a sample)

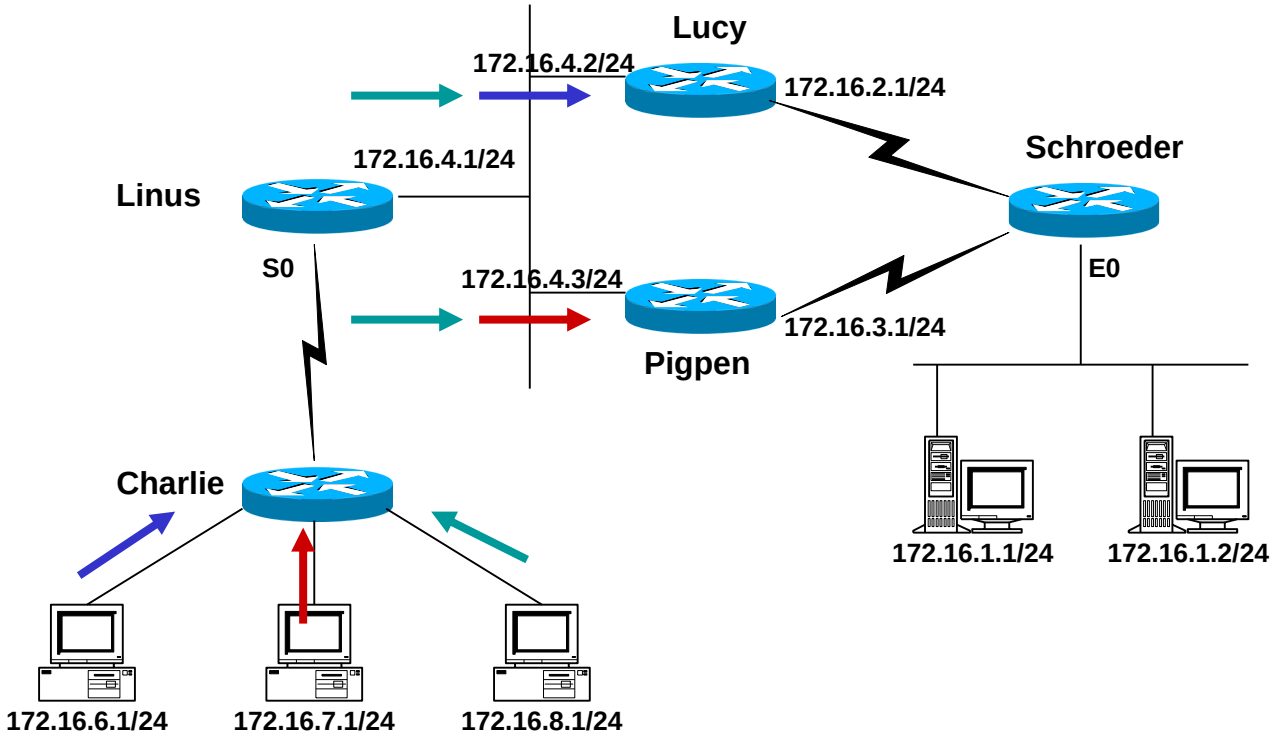
- `Router(config-route-map)#`**`set ip precedence [`*`number`*` | `*`name`*`]`**
  - **Sets precedence value in the IP header. You can specify either the precedence number or name.**
- `Router(config-route-map)#`**`set ip next-hop `*`ip-address`*` [... `*`ip-address`*`]`**
  - **Sets next hop to which to route the packet (the next hop must be adjacent).**
- `Router(config-route-map)#`**`set interface `*`interface-type interface-number`*` [... `*`type number`*`]`**
  - **Sets output interface for the packet.**
- `Router(config-route-map)#`**`set ip default next-hop `*`ip-address`*` [...`*`ip-address`*`]`**
  - **Sets next hop to which to route the packet, if there is no explicit route for this destination.**
- `Router(config-route-map)#`**`set default interface `*`interface-type interface-number`*` [... `*`type ...number`*`]`**
  - **Sets output interface for the packet, if there is no explicit route for this destination.**

Set and Match Options

CCO:

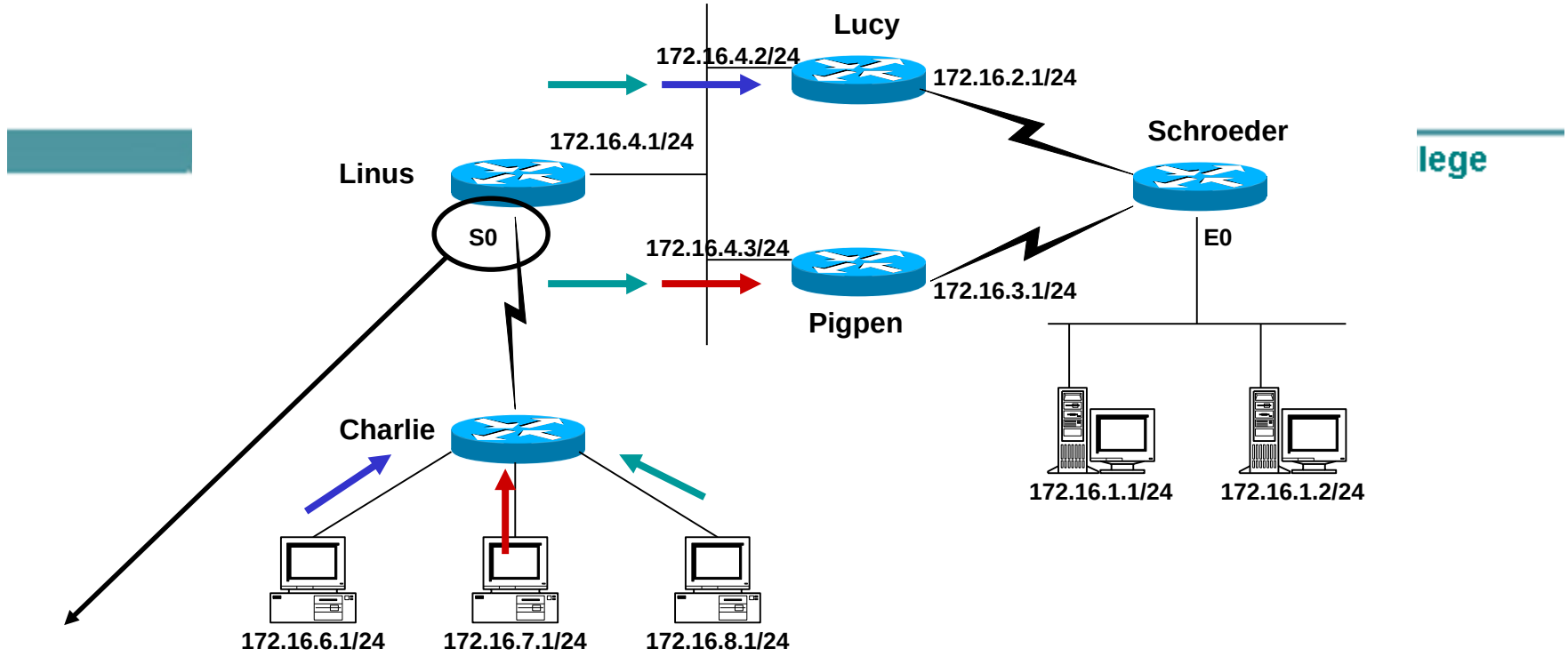
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt1/qcfpbr.htm

Jeff Doyle's Peanuts Example
Single interface example – source IP address



We want to implement a policy on **Linus** such that:

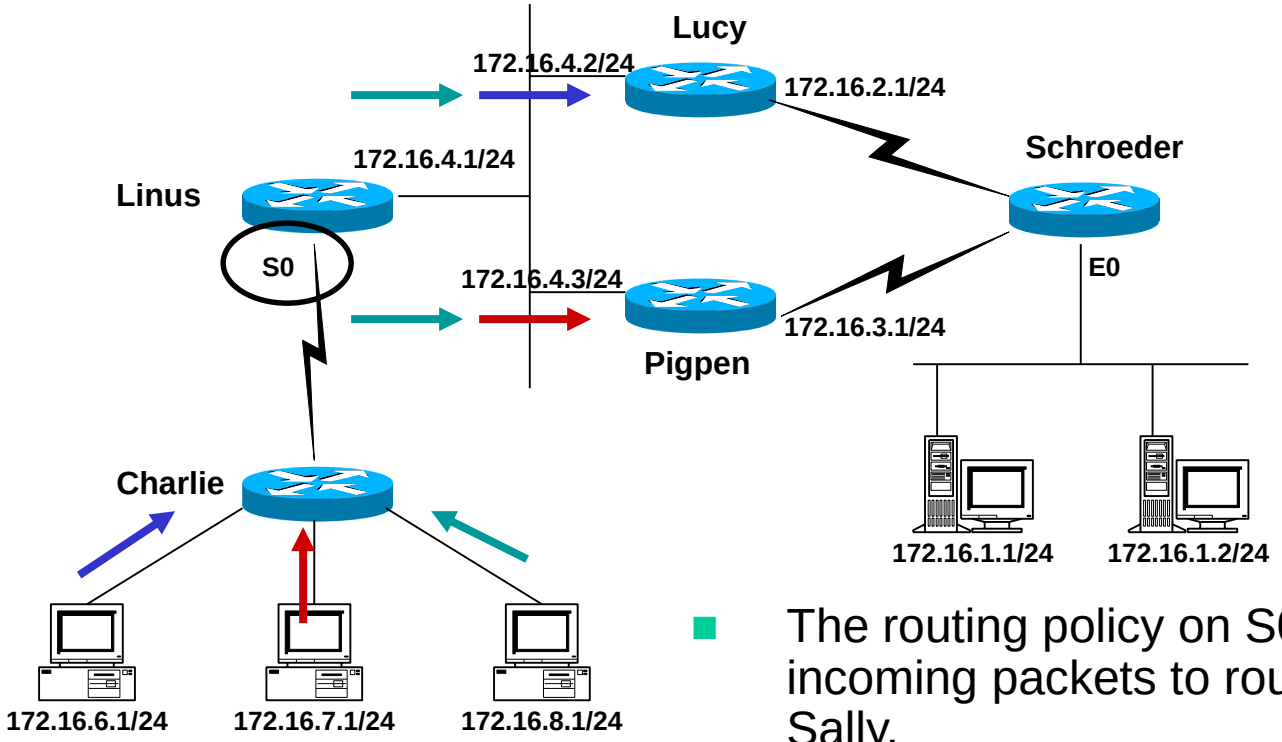
- Traffic from 172.16.6.0/24 subnet is forwarded to Lucy
- Traffic from 172.16.7.0/24 subnet is forwarded to Pigpen
- All other traffic is routed normally



Linus:

```
inter S0
  ip policy route-map Sally

access-list 1 permit
   172.16.6.0 0.0.0.255

access-list 2 permit
   172.16.7.0 0.0.0.255
```

```
route-map Sally permit 10
  match ip address 1
  set ip next-hop 172.16.4.2

route-map Sally permit 15
  match ip address 2
  set ip next-hop 172.16.4.3
```



Linus:

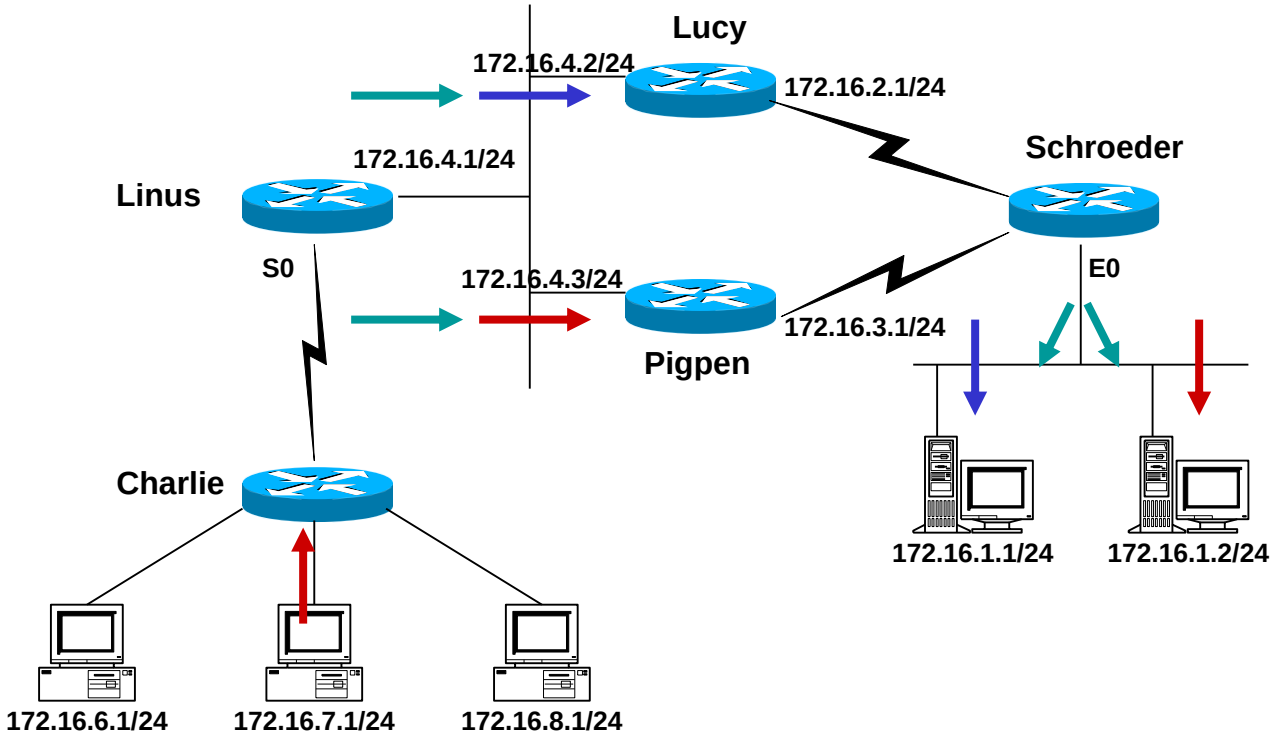
```
inter S0
  ip policy route-map Sally
access-list 1 permit 172.16.6.0 0.0.0.255
access-list 2 permit 172.16.7.0 0.0.0.255

route-map Sally permit 10
  match ip address 1
  set ip next-hop 172.16.4.2
route-map Sally permit 15
  match ip address 2
  set ip next-hop 172.16.4.3
```

- The routing policy on S0 sends incoming packets to route map Sally.
- Statement 10 uses access list 1.
- If a match is made, the packet is forwarded to Lucy.
- If not match is made, the packet is sent to statement 15.
- If a match is made, the packet is forwarded to Pigpen.
- Any packets that do no match 15, such as from 172.16.8.0/24 are routed normally.

Using Extended Access Lists

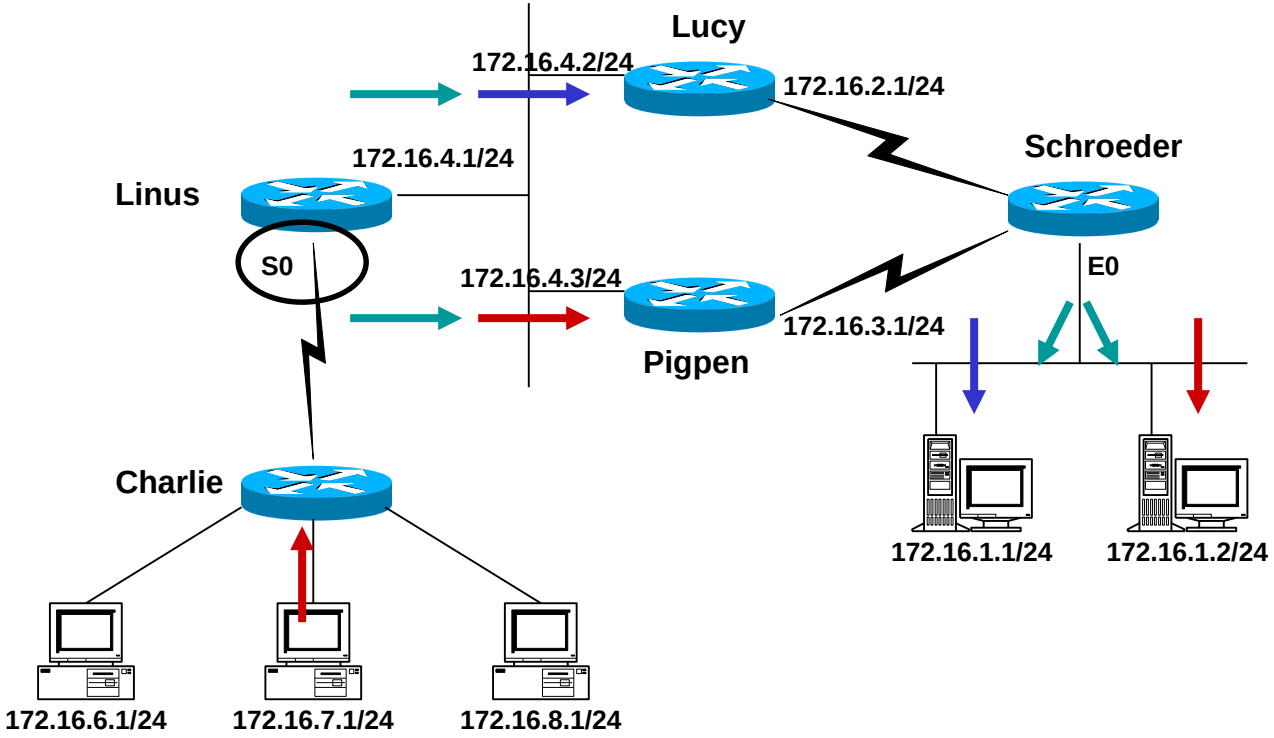
- **`debug ip packet`** can be used to verify the results.
- **Standard access lists** are used when policy routing is by source address only.
- **Extended access lists** are used when policy routing is by both source and destination address.

Jeff Doyle's Peanuts Example
Single interface example – destination IP address



Suppose we want to implement a policy on **Linus** such that:

- Traffic to host 172.16.1.1 is forwarded to Lucy
- Traffic from 172.16.7.1 to host 172.16.1.2 is forwarded to Pigpen
- All other traffic is routed normally



Linus:

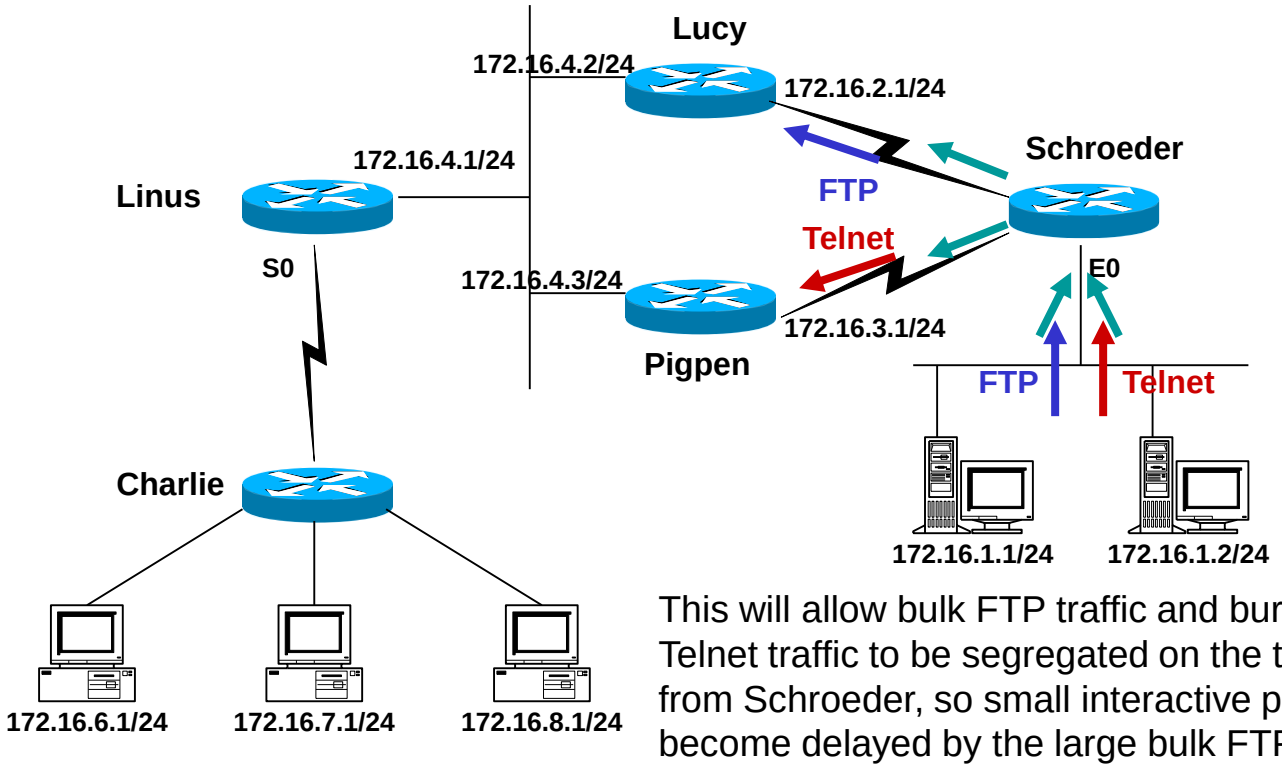
```
inter S0
  ip policy route-map Sally

access-list 101 permit ip any
   host 172.16.1.1
access-list 102 permit ip host
   172.16.7.1 host 172.16.1.2
```

```
route-map Sally permit 10
  match ip address 101
  set ip next-hop 172.16.4.2

route-map Sally permit 15
  match ip address 102
  set ip next-hop 172.16.4.3
```

More Examples

- Let's see more examples, so we can really understand this stuff,…

Jeff Doyle's Peanuts Example
Single interface example – source, destination & port number

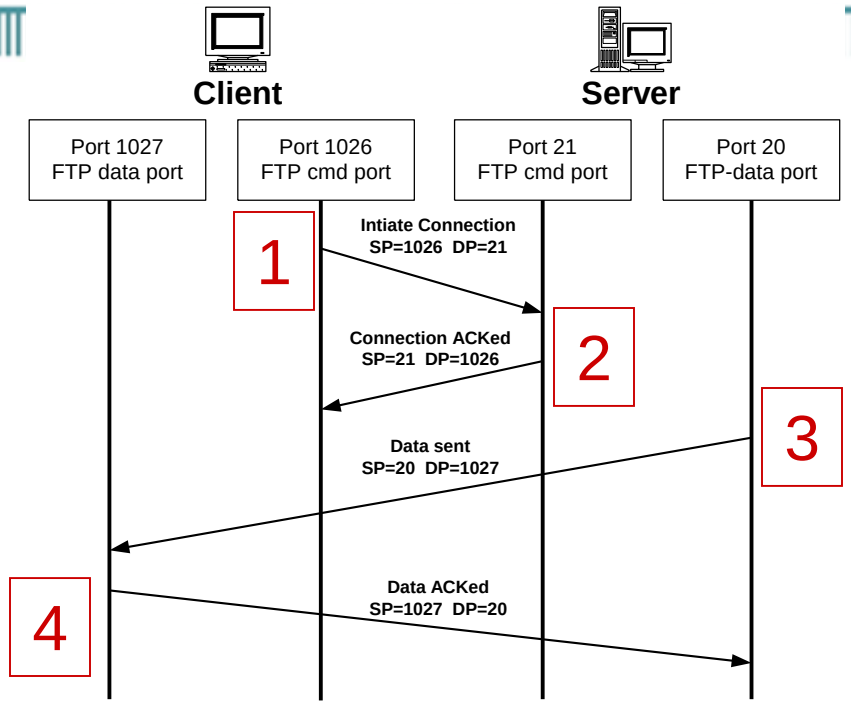


This will allow bulk FTP traffic and bursty, interactive Telnet traffic to be segregated on the two serial links from Schroeder, so small interactive packets do not become delayed by the large bulk FTP packets.

Suppose we want to implement a policy on Schroeder such that:

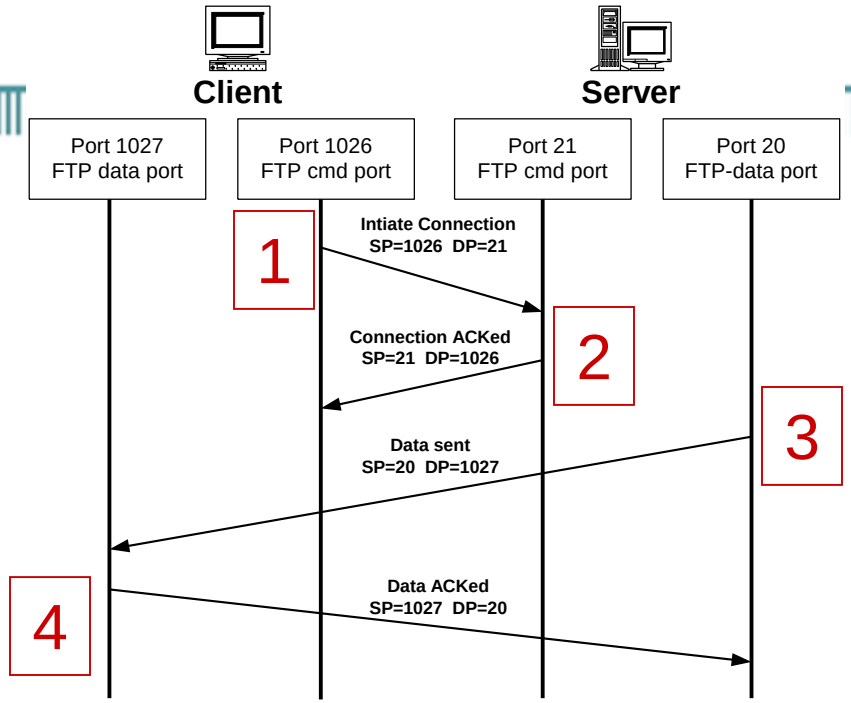
- FTP traffic from 172.16.1.0 servers is forwarded to Lucy
- Telnet traffic from 172.16.1.0 servers is forwarded to Pigpen
- All other traffic is routed normally

A side note on FTP port numbers



Active FTP

- FTP uses different port numbers for initiating a connection and for sending data.
- The FTP client connects from a random unprivileged source port (N > 1024) to the FTP server's command port, destination port 21. **Client: SP=1026 DP=21**
- The FTP client then starts listening to port N+1 (1026+1=1027).
- The server will then connect back to the client's specified data port from its local data port, which is source port 20 to the the client's destination port 1027. **Server SP= 20 DP=1027**

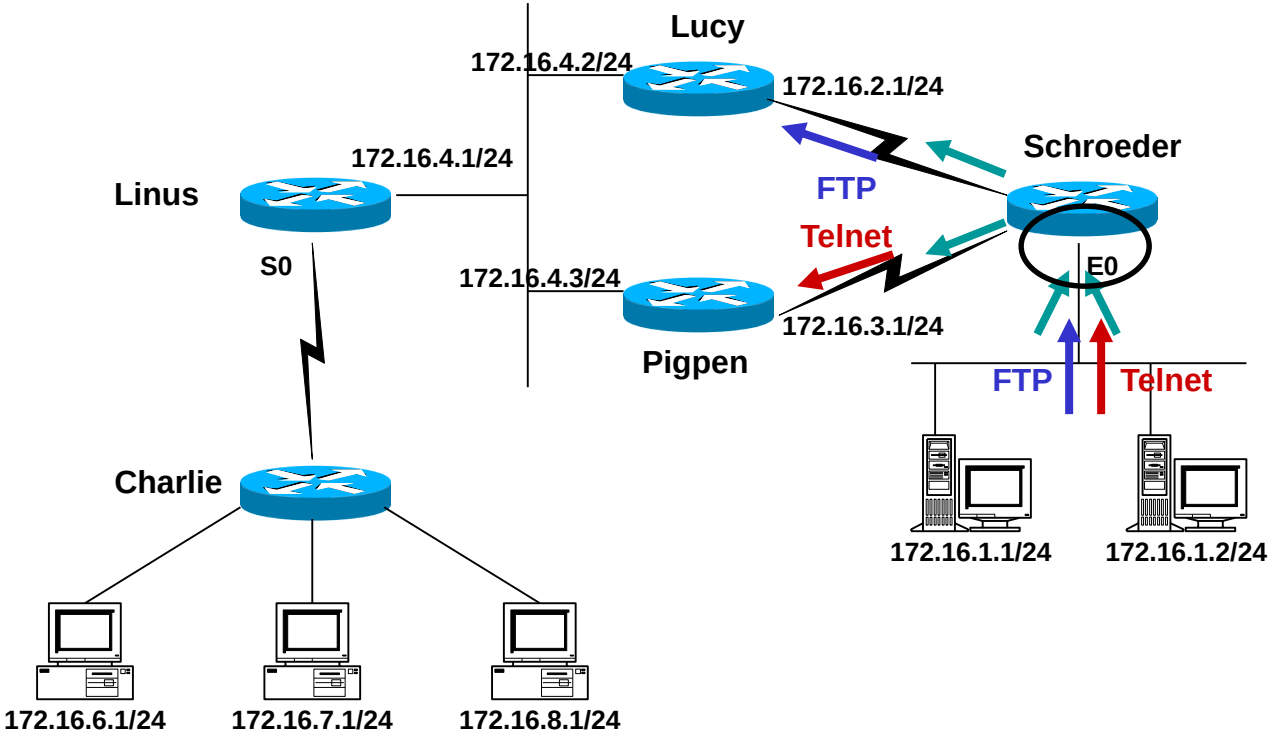


Active FTP

- **Step 1:** The client's command port contacts the server's FTP command port, **Destination port 21 (FTP)**, and **Source port 1026**, sending the command port of 1027 (N+1).
- **Step 2:** The server sends an **ACK** back to the client's command port, using **Source port 21 (FTP)** and **Destination port 1026**.
- **Step 3:** The server initiates a connection on its local data port, **Source port 20 (FTP-data)**, with the **Destination port** set to the FTP data port the client specified earlier as the command port **(N+1) of 1027**.
- **Step 4:** The client sends **ACKs** back (windowing) with the **Source port of 1027** and the **Destination port of 20 (FTP-data)** .

For more on FTP…

- **Active FTP vs. Passive FTP, a Definitive Explanation**
- **http://slacksite.com/other/ftp.html**



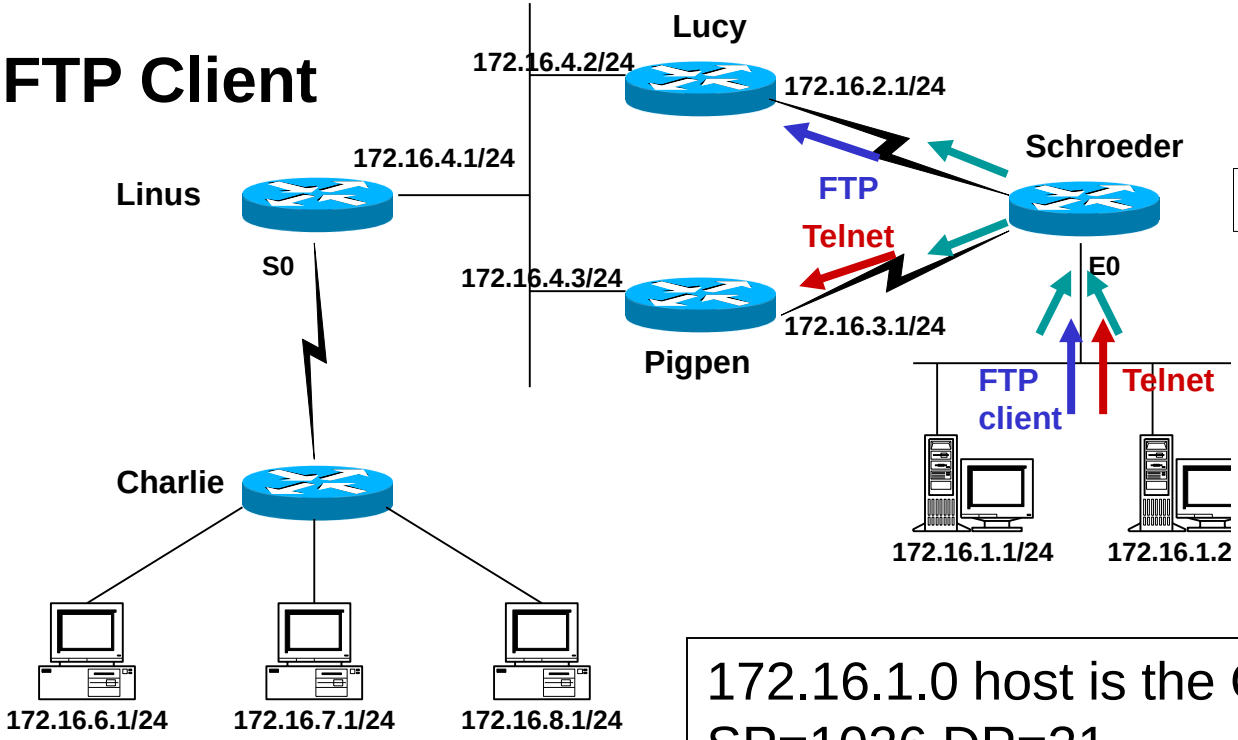
Note on ACLs: The "ftp" following the "source ip" refers to the source port, whereas the "ftp" after the "destination ip" refers to the "destination port." The client uses destination port 21 (FTP) for sending (control) data to the server, whereas the server uses source port 20 (FTP-DATA), instead of port 21 when sending data back to the client. The server uses a port > 1024 when FTP is done in "passive mode," thus use "ip access-list 105 permit any 172.16.1.0 0.0.0.255 established"

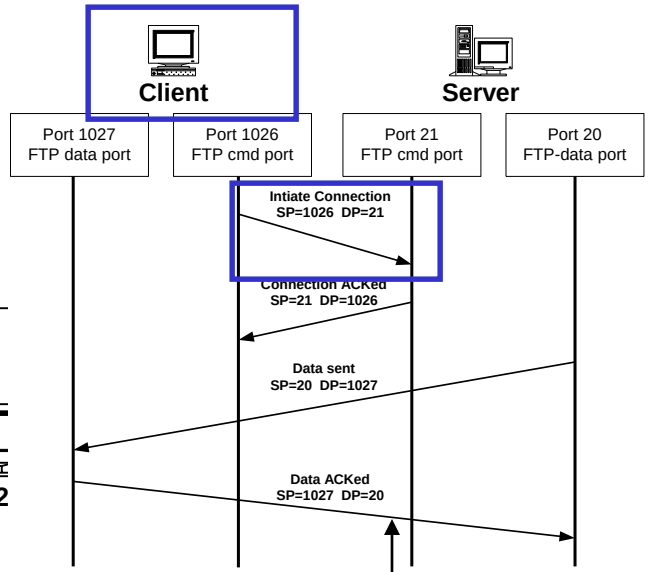
Schoeder

```
inter E0
  ip policy route-map Rerun
! Used when 172.16.1.1 is the client
access-list 105 permit tcp 172.16.1.0
    0.0.0.255 any eq ftp
! Used when 172.16.1.1 is the server
access-list 105 permit tcp 172.16.1.0
    0.0.0.255 eq ftp-data any
access-list 106 permit tcp 172.16.1.0
    0.0.0.255 eq telnet any
```

```
route-map Rerun permit 10
  match ip address 105
  set ip next-hop 172.16.2.1

route-map Rerun permit 20
  match ip address 106
  set ip next-hop 172.16.3.1
```

FTP Client





172.16.1.0 host is the Client:
SP=1026 DP=21

Match DP=21 (FTP)

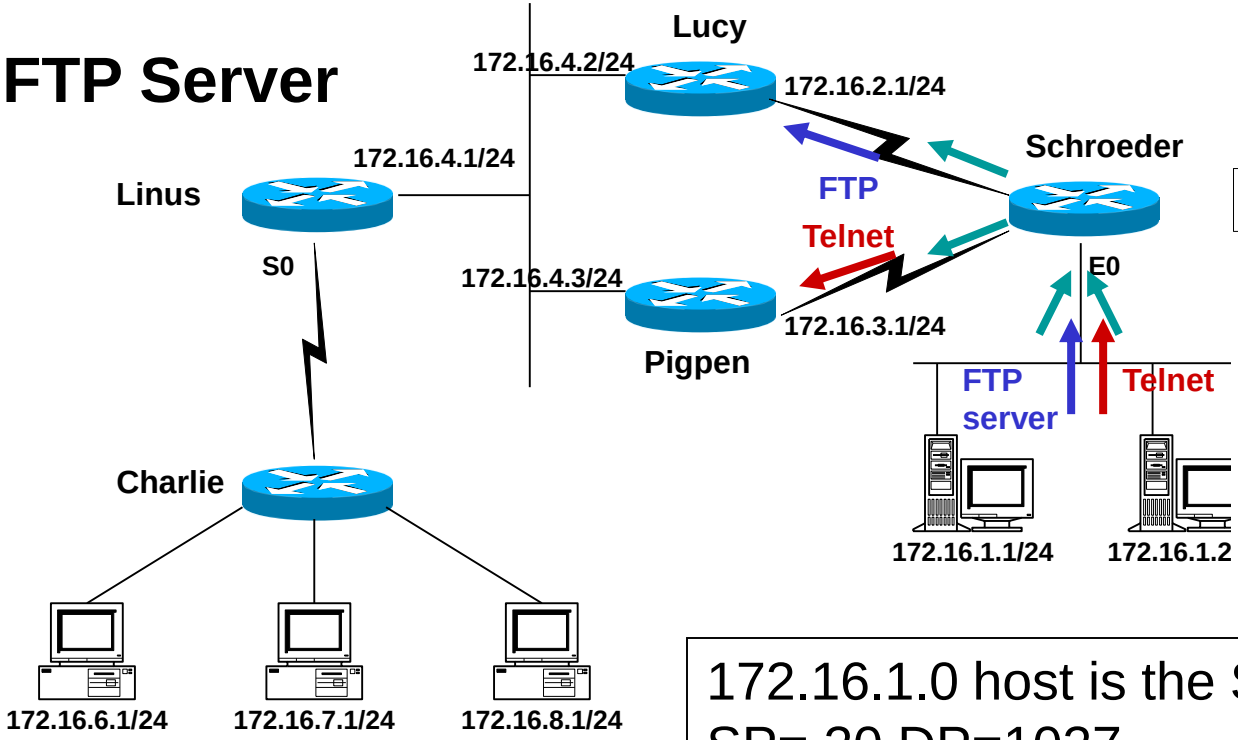
What about these?

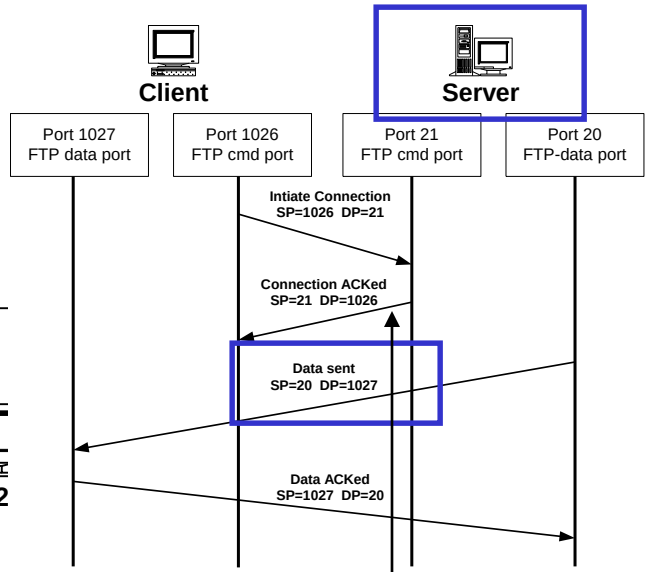
Schoeder

```
inter E0
  ip policy route-map Rerun
! When 172.16.1.h is the client
! This means 172.16.1.h DP eq ftp

access-list 105 permit tcp 172.16.1.0
    0.0.0.255 any eq ftp

access-list 106 permit tcp 172.16.1.0
    0.0.0.255 eq telnet any
```

```
route-map Rerun permit 10
  match ip address 105
  set ip next-hop 172.16.2.1

route-map Rerun permit 20
  match ip address 106
  set ip next-hop 172.16.3.1
```

FTP Server





172.16.1.0 host is the Server:
SP= 20 DP=1027

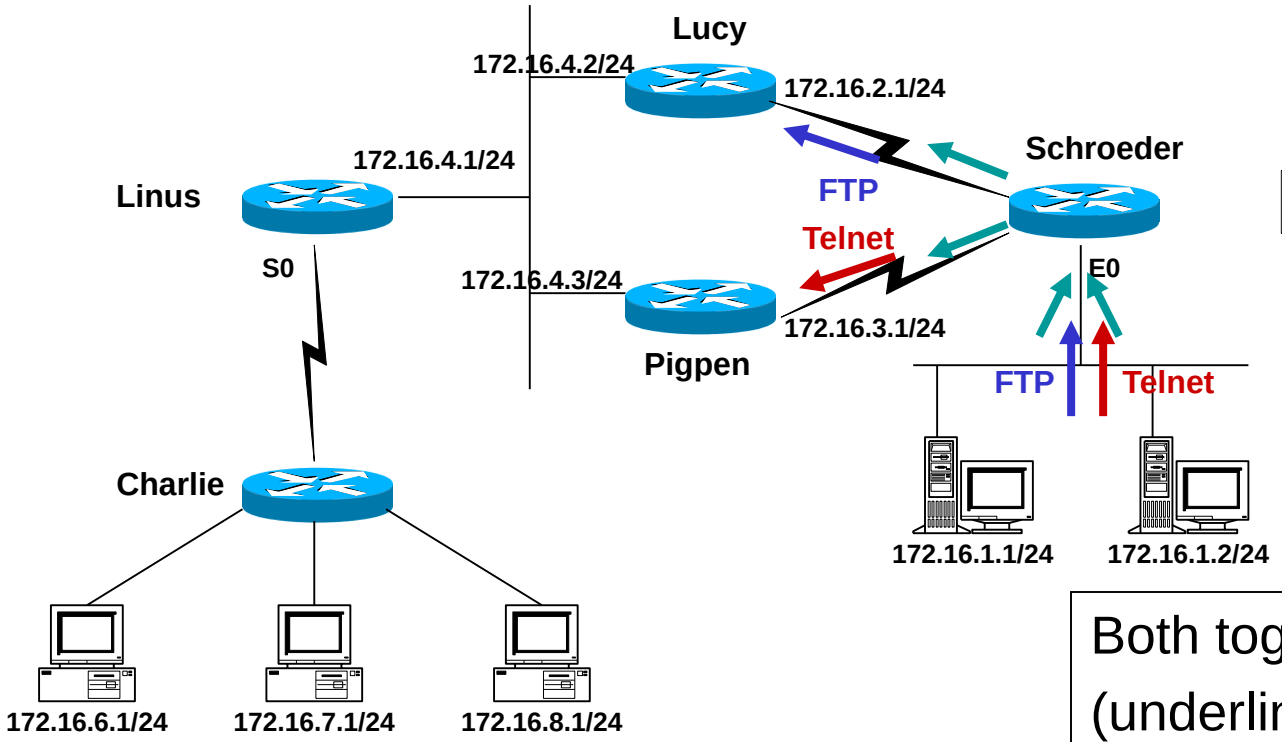
Match SP=20 (FTP-data)

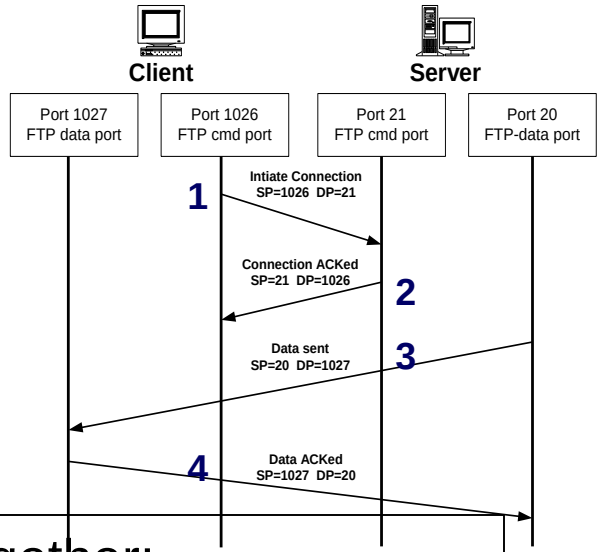
What about this?

Schoeder

```
inter E0
  ip policy route-map Rerun
! When 172.16.1.h is the server
! This means 172.16.1.h SP eq ftp-data

access-list 105 permit tcp 172.16.1.0
    0.0.0.255 eq ftp-data any

access-list 106 permit tcp 172.16.1.0
    0.0.0.255 eq telnet any
```

```
route-map Rerun permit 10
  match ip address 105
  set ip next-hop 172.16.2.1

route-map Rerun permit 20
  match ip address 106
  set ip next-hop 172.16.3.1
```



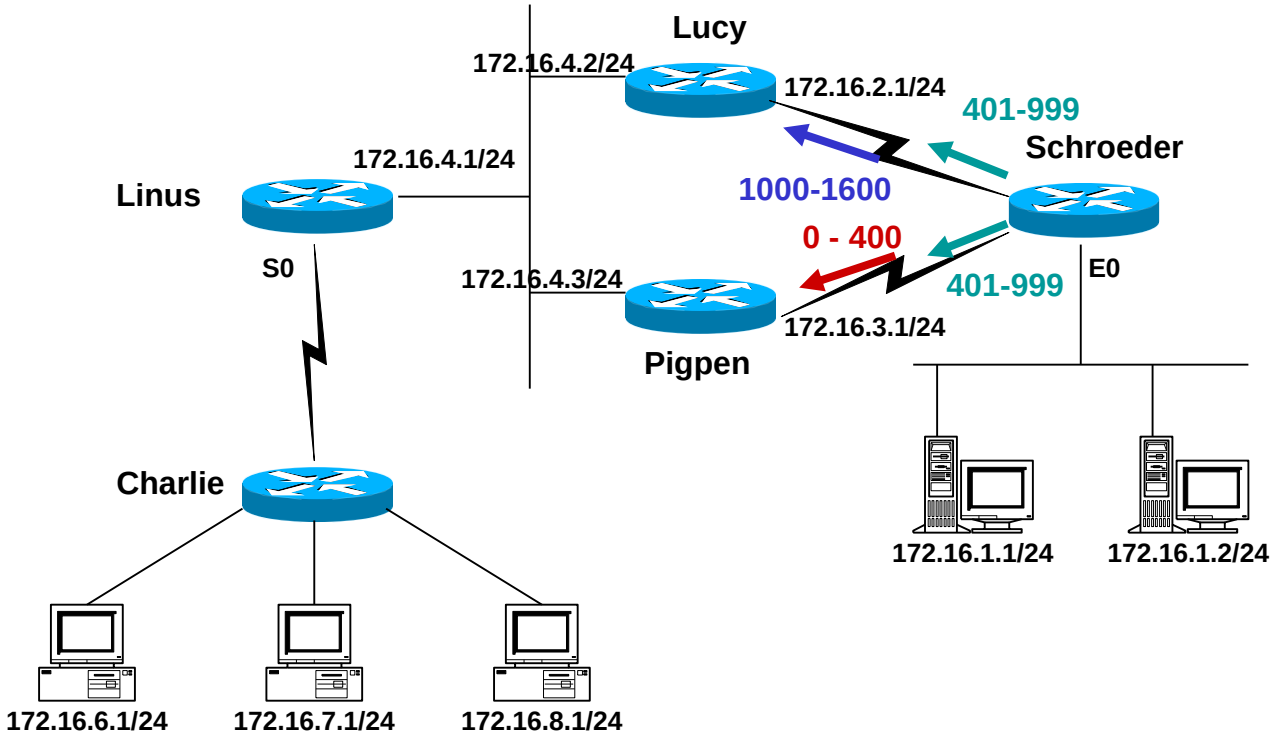


Both together:
(underline is checked)
Client: SP=1026 DP=21
Server: SP= 20 DP=1027

```
inter E0
  ip policy route-map Rerun
! Used when 172.16.1.1 is the client
1 access-list 105 permit tcp 172.16.1.0 0.0.0.255
      any eq ftp
4 access-list 105 permit tcp 172.16.1.0 0.0.0.255
      any eq ftp-data
! Used when 172.16.1.1 is the server
3 access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq
      ftp-data any
2 access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq
      ftp any
  access-list 106 permit tcp 172.16.1.0 0.0.0.255 eq
      telnet any
```

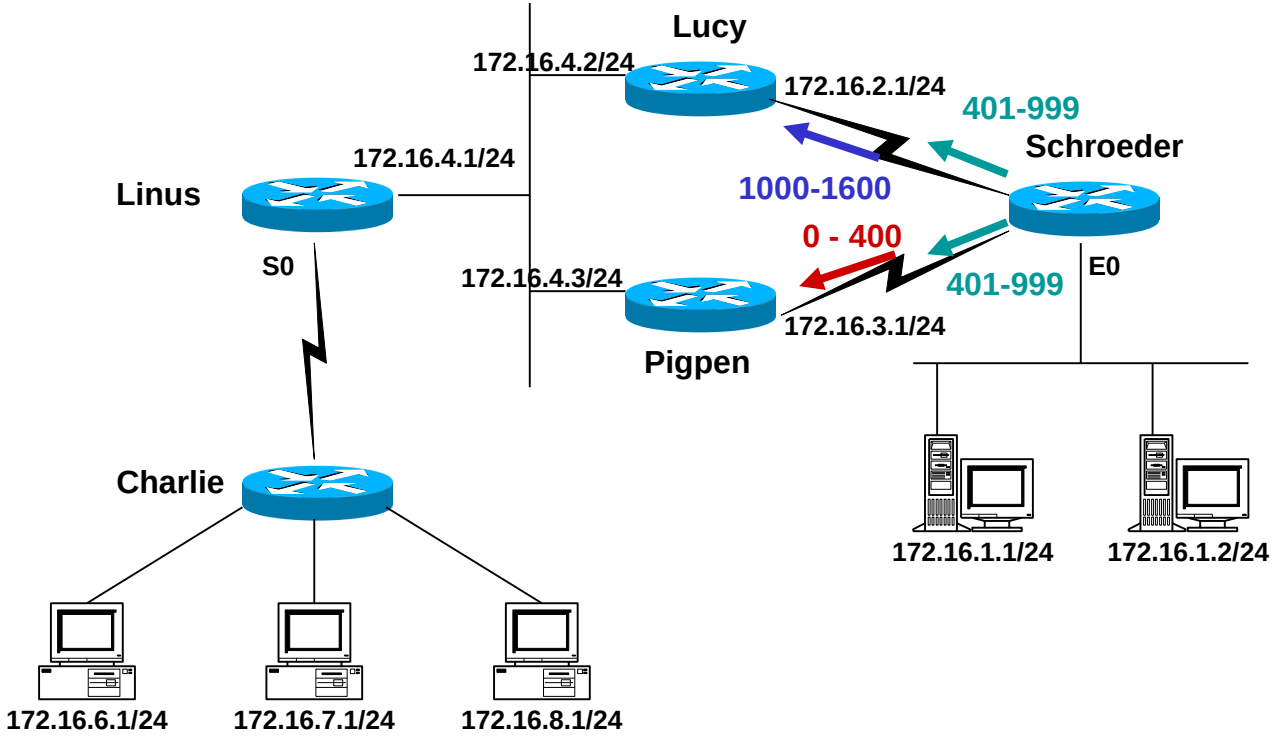
```
route-map Rerun permit 10
  match ip address 105
  set ip next-hop 172.16.2.1

route-map Rerun permit 20
  match ip address 106
  set ip next-hop 172.16.3.1
```

Jeff Doyle's Peanuts Example
Single interface example – match length



Suppose we want to implement a policy on **Schroeder** such that:

- All packets between 1000 and 1600 bytes are forwarded to Lucy
- All packets up to 400 are forwarded to Pigpen
- All other traffic, packets between 401 and 999, are routed normally



Schoeder

```
inter E0
  ip policy route-map Woodstock
```

```
route-map Woodstock permit 20
  match length 1000 1600
  set ip next-hop 172.16.2.1

route-map Woodstock permit 30
  match length 0 400
  set ip next-hop 172.16.3.1
```

End of Part 1

Any Questions?

Next week…
Route Optimization

- Passive Interfaces
- Route Filters
  - Distribute Lists
- Policy Routing
  - Route Maps
- Route Redistribution
  - Multiple Routing Protocols
  - Changing Administrative Distances
  - Configuring Redistribution
  - Default Metrics